Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Trabajo Fin de Grado

Aplicación de realidad virtual para mostrar datos de monitorización en 3D capturados en entornos reales

Virtual Reality application to display 3D monitoring data acquired in real environments

Autora: Beatriz Salvador Ramos

Directores:

Ana Cristina Murillo Arnal

Alejandro Rafael Mosteo Chagoyen

Ingeniería Electrónica y Automática
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Agosto 2019

Resumen

La realidad virtual (RV) está cada vez más presente en la sociedad. Ha experimentado un gran desarrollo en los últimos años y cada vez se utiliza en más y muy diferentes aplicaciones, ya que tiene un gran potencial.

El objetivo en este proyecto es la puesta en marcha de un *headset* de realidad virtual, su uso y la implementación de dos aplicaciones demostrativas para dicho *headset* donde se incluyan datos 3D capturados en entornos reales. La idea es tener dos aplicaciones que sirvan de base para facilitar la divulgación, demostración y uso de modelos capturados en el grupo de investigación donde se realiza el proyecto.

El primer aspecto a abordar para la realización del proyecto es la selección del hardware y del software necesario. En el caso del hardware se ha utilizado el *headset* de realidad virtual *HTC Vive Pro*, para la visualización de las aplicaciones e interacción con ellas, y un *Structure sensor* acoplado a un iPad para obtener modelos 3D de zonas y objetos de la universidad. Como software para la programación de las aplicaciones, se ha decidido utilizar el motor gráfico *Unreal Engine 4* ya que actualmente es uno de los más utilizados con versión gratuita y con multitud de opciones de programación. Además del motor es necesario el software específico para Vive Pro, *SteamVR*.

Tras esta elección, en el proyecto también se ha cubierto desde la conexión y calibrado del hardware, al estudio del software y realización de los programas demostradores. Se estudia tanto la manera de importar los datos recogidos de entornos reales como la forma de animar estos modelos y elementos virtuales e interactuar con todo el escenario ya sea mediante el ratón, el teclado o los controladores propios del *set* de realidad virtual, investigando a fondo el funcionamiento del *set* completo (visor, mandos y estaciones base) y la programación en el motor *Unreal Engine 4*. Puesto que el *headset* HTC Vive Pro es de los más nuevos en el mercado, no se había usado antes en el grupo de investigación y se parte de cero en la realización de este proyecto.

Como resultado de este proyecto, se han programado dos aplicaciones de realidad virtual que utilizan todas las características principales del entorno elegido. En los demostradores realizados se procesan y se visualizan datos capturados con distintos sensores de visión 3D, combinándolos con elementos virtuales. En primer lugar se ha realizado una aplicación recreando un fondo submarino real, modelado a partir de un mosaico de nubes de puntos capturados con una cámara submarina en un proyecto de investigación real. El segundo tipo de datos utilizados son modelos de mallas 3D, construidos desde una cámara RGB-d durante este proyecto. Se utilizan tanto imágenes del entorno, simulando datos tomadas durante el recorrido realizado por un robot incorporando en el una cámara o sensor adecuado, como imágenes de cualquier objeto cotidiano capturado con cámaras RGB-d e insertado como elemento virtual.

Estas dos aplicaciones sirven de demostrador para distintos proyectos de investigación y para sentar una base que se puede utilizar en trabajos posteriores relacionados.

Índice general

Índice de figuras

Capítulo 1

Introducción

Este proyecto presenta dos demostradores implementados para visualizar en unas gafas de realidad virtual datos en 3D tomados en entornos reales. A continuación se va a exponer la motivación y el contexto que han impulsado la realización de este trabajo, así como los objetivos que se esperan obtener y las tareas llevadas a cabo en el mismo.

Figura 1.1: Distintos dispositivos relacionados con robótica y realidad virtual.

1.1. Motivación

El creciente desarrollo de la robótica y de la realidad virtual han motivado la realización de este trabajo. El uso de estas tecnologías, de las cuales se puede ver algún ejemplo en la Figura 1.1, se está implantando cada vez más en la sociedad de muy diversas formas y con múltiples aplicaciones.

La realidad virtual, aumentada y mixta se utiliza en sectores profesionales como puede ser en la industria, en empresas para mejorar sus operaciones internas o incluso en los quirófanos y hospitales. Ejemplo de una aplicación en

alguno de estos sectores es el sistema HoloSurg creado por Exovite[1], que utiliza las gafas HoloLens[2] y realidad mixta para mejorar la seguridad y precisión en cirugías. Pero el alcance de estas tecnologías no se queda solo aquí, actualmente hay usos mucho más cotidianos de estas como puede ser en el mundo de los videojuegos o en museos. En estos últimos, se pueden recrear escenarios inaccesibles para las personas o de otras épocas para poder mostrarlos al público y además hacerlo de manera inmersiva e interactiva para captar así una mayor atención. Esto puede ayudar a promover un mayor interés en ellos. Un ejemplo cercano del uso de la realidad virtual en los museos es el proyecto «Entrar en el cuadro»[3] que se ha realizado en el museo Thyssen-Bornemisza, donde los visitantes pueden adentrarse y recorrer las escenas plasmadas en los cuadros de manera realista. Además debido a su éxito, esta iniciativa recorrerá más ciudades españolas este año, incluyendo Zaragoza. En la Figura 1.2 se pueden observar las dos aplicaciones mencionadas. Del mismo modo, la realidad virtual se puede aplicar al ámbito de la enseñanza ya que puede ayudar a promover la motivación por el aprendizaje entre los más jóvenes.

(a) (b)

Figura 1.2: Ejemplos de algunas aplicaciones de realidad virtual actualmente. (a) Gafas HoloLens y su uso en hospitales (HoloSurg). (b) Proyecto con realidad virtual «Entrar en el cuadro» en el museo Thyssen-Bornemisza.

Como se puede apreciar en lo expuesto anteriormente, la realidad virtual supone uno de los cambios tecnológicos más importantes de los últimos tiempos. Se trata de la «representación de escenas o imágenes de objetos producida por un sistema informático, que da la sensación de su existencia real» según la Real Academia Española[4]. Con ella se puede entrar a formar parte de un mundo ficticio, que puede estar basado en entornos reales o no, a través de unas gafas de realidad virtual.

Actualmente está experimentando un gran crecimiento, pero tiene su origen entre los años 1957 y 1962 con Sensorama[5]. En la misma época se empiezan a desarrollar unos dispositivos como *Headsight*, que se puede ver en la Figura 1.3a, y *Ultimate Display* o "La Espada de Damocles"que serán los predecesores a las gafas de realidad virtual que se conocen hoy en día. En 1993 SEGA desarrolla unas gafas de realidad virtual que anticipan el formato que terminará

[1] `http://www.exovite.com/es/exovite-es/`
[2] `https://www.microsoft.com/es-es/hololens`
[3] `https://www.museothyssen.org/apoyo/apoyo-empresarial/entrar-cuadro-realidad-virtual-itinerancia2019`
[4] `https://dle.rae.es/?id=VH7cofQ`
[5] `https://proyectoidis.org/sensorama/`

triunfando finalmente, mostradas en la Figura 1.3b. A partir de entonces y hasta hoy se han ido desarrollando *softwares* específicos y se ha ido mejorando la calidad y comodidad del *hardware*, teniendo ahora multitud de opciones como pueden ser Oculus Rift[6], Samsung Gear VR[7], HTC Vive Pro[8] que son las utilizadas para este trabajo, las Microsoft HoloLens ya mencionadas anteriormente o Google Cardboard [9] que pretende proporcionar una experiencia de realidad virtual asequible para todo el mundo. La comparativa del primer prototipo de *head-mounted display* (HMD) de 1961 con las SEGA VR y con las Vive Pro usadas en este proyecto se puede ver en la Figura1.3.

(a) (b) (c)

Figura 1.3: Evolución de las gafas de realidad virtual. (a) *Headsight*, primer prototipo de *head-mounted display*. (b) Formato de gafas de realidad virtual desarrollado por SEGA en 1993. (c) *Headset* completo HTC Vive Pro.

La robótica es una disciplina en constante desarrollo que actualmente evoluciona hacia los sistemas móviles, dotándolos cada vez de más autonomía. Esto hace que aunar este campo con el de la realidad virtual cobre interés. Por ello este proyecto busca visualizar datos 3D tomados en entornos reales desde alguno de estos sistemas ampliamente utilizados en el campo de la robótica, integrándolos en aplicaciones diseñadas para distintos propósitos.

En este trabajo se han desarrollado dos demostradores de realidad virtual para dos aplicaciones distintas. Uno para la navegación por un espacio de interiores capturado por sistemas inteligentes que se puede utilizar para tareas de inspección y mantenimiento y otro para la visualización de modelos capturados en zonas de difícil acceso, en este caso de corales del fondo submarino, que después se pueden mostrar por ejemplo en museos.

1.2. Trabajo relacionado

Este proyecto se centra en el estudio de la realidad virtual y el desarrollo de aplicaciones para ella gracias a motores gráficos preparados para este propósito.

Para ello es necesario realizar un estudio sobre las formas de importar y manipular los datos dentro del motor gráfico elegido, ***Unreal Engine* 4**[10], ampliamente utilizado para el desarrollo de videojuegos y de aplicaciones de realidad

[6] `https://www.oculus.com/rift/?locale=es_ES`
[7] `https://www.samsung.com/es/wearables/gear-vr-sm-r325nzvaphe/`
[8] `https://www.vive.com/mx/product/vive-pro/`
[9] `https://vr.google.com/cardboard/`
[10] `https://www.unrealengine.com/en-US/`

virtual. Para aprender el uso de este programa se ha utilizado la documentación oficial de *Unreal*[1].

Además se ha estudiado en detalle qué es la **realidad virtual**, su funcionamiento e historia, las tecnologías que utiliza y algunas de sus aplicaciones. Un resumen sobre estos temas lo encontramos en el artículo «*Overview of Virtual Reality Technologies*»[2].

Después de las nociones generales el estudio se ha centrado en los ***headsets*** de realidad virtual existentes en la actualidad, sobre todo en el HTC Vive Pro[11], para lo que se ha usado su guía de usuario[3]. Además en el artículo «*The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research*»[4] se analiza de manera cuantitativa la precisión del *headset* HTC Vive, que es la versión anterior del utilizado cuyas características son similares o incluso iguales en algunos casos y su funcionamiento sigue los mismos principios.

Aplicaciones. Los demostradores implementados en este proyecto son una aplicación para visualizar y navegar por un espacio de interiores y la recreación de un escenario de un fondo submarino. Esta última permite la visualización y conocimiento de las especies que allí viven, por lo que se puede incorporar su uso en el ámbito de la divulgación científica, pudiéndose utilizar en museos o en el campo de la enseñanza. Otros proyectos que tratan la realidad virtual como ayuda para la enseñanza son «*Realidad Aumentada en la Educación: una tecnología emergente*»[5] y «*Science learning in virtual environments: a descriptive study*»[6]. Recientemente también se está implantando su uso en museos, lo cual se trata en « *Virtual reality interactivity in a museum environment*»[7], e incentivando su utilización para ayudar a la difusión y protección del patrimonio cultural. Ejemplo de esto es el proyecto ROVINA[8], que ha desarrollado robots que accedan a las catacumbas de Santa Priscila (Roma), un lugar contaminado por gases. Así se permite la visita virtual al mismo y se ayuda al conocimiento y mantenimiento de lugares con gran valor pero no accesibles.

Finalmente, una de las aplicaciones en las que más se está intentando introducir la realidad virtual y aumentada actualmente es en la medicina, donde puede suponer un gran avance. En el reciente artículo publicado este mismo año «*Applications of Virtual and Augmented Reality in Biomedical Imaging*»[9] se habla del uso de estas tecnologías para una mejor planificación y monitorización de las cirugías, utilizando modelos tridimensionales que se pueden visualizar y manipular.

1.3. Contexto de realización del trabajo

Este proyecto se ha realizado en el grupo de investigación *Robotics, Perception and Real Time (RoPeRT)*, dentro del Instituto de Ingeniería e Investigación de Aragón (I3A). Se ha desarrollado bajo la supervisión de la directora de este grupo y la ayuda y experiencia de otros integrantes de él.

El objetivo es crear una aplicación que sirva de demostrador para futuros proyectos. A pesar de que dentro del grupo no se había realizado ningún trabajo ni con el *headset* HTC Vive Pro ni con el software *Unreal Engine* y por tanto

[11] `https://www.vive.com/mx/product/vive-pro/`

se partía desde cero en la creación del programa, se han podido utilizar datos de un proyecto anterior. Concretamente se usan algunos modelos de mosaicos de nubes de puntos de corales para la aplicación en la que se recrea un fondo submarino. Muchos de ellos se encuentran en Sketchfab[12].

Desde un punto de vista personal, gracias a trabajar en este laboratorio he aprendido a utilizar nuevos programas y herramientas como el motor *Unreal Engine* 4, el *headset* de realidad virtual HTC Vive Pro o el *Structure Sensor*, pero también he visto como se trabaja dentro de un equipo, algo muy importante de cara al futuro.

1.4. Objetivos y tareas

El objetivo principal de este proyecto es la puesta en marcha de un *headset* de realidad virtual, el HTC Vive Pro y aprender tanto su manejo como el del software *Unreal Engine*, para el diseño e implementación de dos aplicaciones que permitan visualizar/procesar diferentes tipos de datos 3D capturados en entornos reales. Estos datos serán capturados con distintos dispositivos típicos en aplicaciones de robótica, como pueden ser distintos sensores de visión o información 3D. Se quiere poder mostrar y analizar los escenarios creados a partir de los datos reales anotando el contenido de los mismos con elementos virtuales con los que además se pueda interactuar.

Los problemas abordados en este trabajo son: la puesta en marcha del *hardware* (*headset* de realidad virtual HTC Vive Pro) y del entorno *software* (elección del SO, del entorno adecuado y uso del mismo); la captura e importación de datos compatibles con el entorno elegido; y por último la programación de una aplicación de realidad virtual que sirva de demostrador para distintos proyectos de investigación.

Dentro de las tareas realizadas se encuentran:

- La instalación de todos los programas necesarios y estudio del manejo/programación del software elegido, *Unreal Engine*, realizando pequeños tutoriales.
- El estudio y puesta en marcha del *headset* que se va a utilizar (HTC Vive Pro), para lo cual se han estudiado los requisitos del *hardware*.
- Un estudio de las técnicas para importar y convertir los datos capturados hasta poder programar y construir escenarios en una aplicación de realidad virtual. Los datos se capturan al menos desde 2 sistemas/fuentes diferentes utilizadas en proyectos de investigación del grupo en el que se desarrolla el proyecto: modelos de mallas 3D construidos desde una cámara RGB-d (*Structure sensor*) durante la realización del trabajo y modelos de mosaicos de nubes de puntos capturados con una cámara submarina en un proyecto de investigación real.
- Un estudio teórico-práctico de opciones de visualización de los datos y decidir los distintos elementos virtuales que puedan interactuar con dichos escenarios.

[12] `https://sketchfab.com/Marine_Imaging_Lab`

- Desarrollo de los dos demostradores. Se ha utilizado el motor *Unreal Engine* 4 para su diseño y programación, para lo cual ha sido necesario leer la documentación oficial proporcionada por *Unreal* complementada con otras fuentes de información y aprender su manejo.
- Análisis sobre el potencial y las posibles mejoras de los demostradores.
- La elaboración de la documentación.

Distribución temporal. La distribución temporal de cada una de las tareas llevadas a cabo durante la realización del trabajo se puede ver en la Figura 1.4 de manera aproximada. En la imagen cada fila representa una tarea y cada columna un mes. Las columnas están divididas en dos de forma que quede separado en cada mes la primera quincena de la segunda.

1.5. Contenido de la memoria

En el capítulo 2 se explican los tipos de modelos utilizados, la manera en la que se capturan y la forma de importarlos al *software* utilizado. El capítulo 3 describe el sistema utilizado para la creación de los demostradores, analizando las alternativas en el mercado y centrándose en la opción elegida entre ellas. Se explica tanto la parte *hardware* y su puesta en marcha como la parte de *software* y sus herramientas y características destacadas. En el capítulo 4 se presentan los demostradores desarrollados, explicando además su propósito y aplicación. Finalmente, el capítulo 5 recoge las conclusiones extraídas durante la realización del proyecto.

Como material adicional se incluyen dos anexos. El anexo A contiene especificaciones, la explicación del funcionamiento básico del sistema de seguimiento y un manual paso a paso de conexión del *headset* elegido para el proyecto. En el anexo B se explica en detalle una de las características más interesantes estudiadas del *software* utilizado, la creación de una trayectoria predefinida a seguir por un personaje en un escenario.

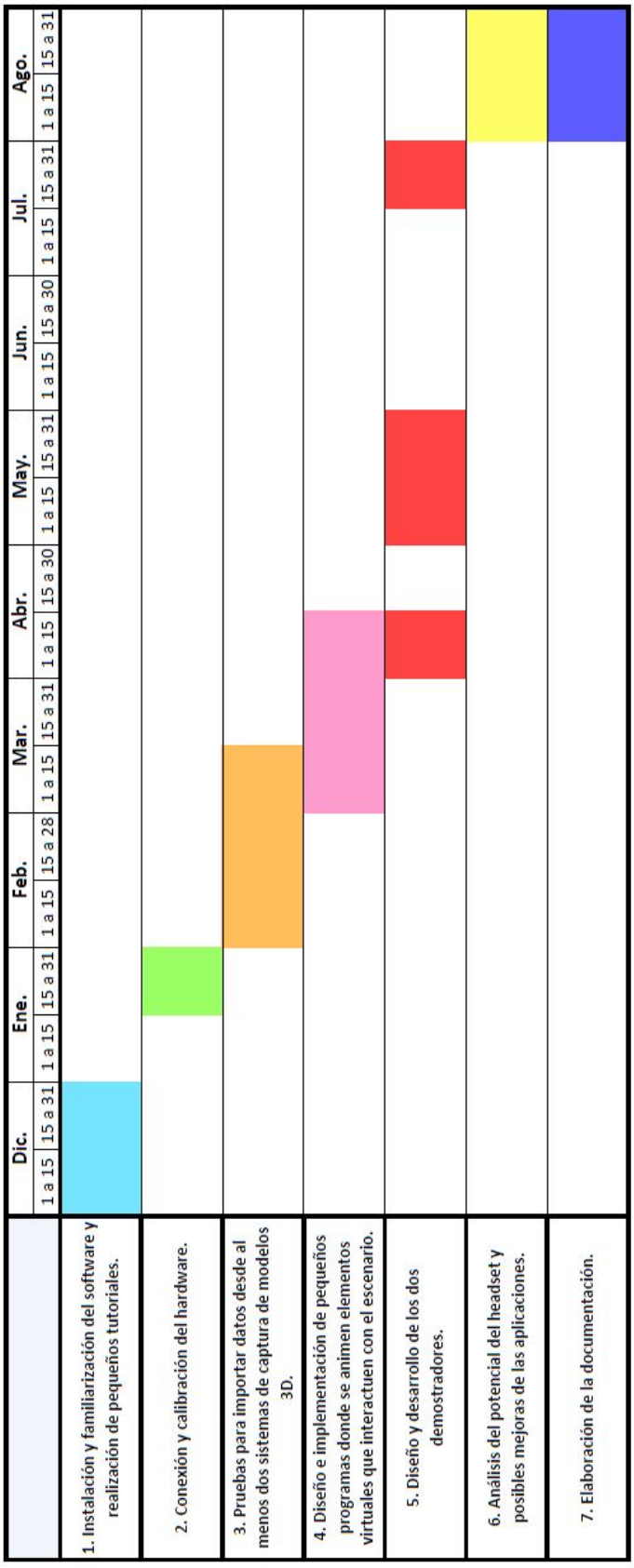


Figura 1.4: Diagrama de Gantt aproximado de la distribución de tareas llevadas a cabo en el proyecto

Capítulo 2

Modelos 3D

Este capítulo describe los distintos tipos de modelos 3D utilizados para llevar a cabo el proyecto. Se trata tanto su obtención como la manera de importarlos al motor gráfico *Unreal Engine* 4. En este trabajo se utilizan dos tipos de datos diferentes, tomados desde distintas fuentes. El primer tipo, descrito en la sección 2.1, consiste en modelos de nubes de puntos, de los que se puede ver un ejemplo en la Figura 2.1. El segundo tipo de datos utilizados en este proyecto son los modelos de mallas 3D que se explican en la sección 2.2.

fuente: `https://dronica.es/nube-de-puntos-que-es-y-para-que-sirve/`

Figura 2.1: Modelo 3D de nube de puntos de la Catedral de Murcia tomado con el escáner láser Leica RTC360.

2.1. Modelos 3D a partir de nubes de puntos

Una **nube de puntos 3D** se compone de millones de puntos posicionados en el espacio, formando una entidad física y representando la superficie externa de un objeto. Las nubes de puntos contienen información, tanto dimensional y geométrica como de la reflectividad y color, de las superficies escaneadas.

Se pueden obtener de manera directa a través de un escáner láser o de la fotogrametría digital.

Sensores de profundidad. La primera forma, el escáner láser 3D, es un dispositivo de adquisición de datos masivos ya que mide de forma automática un gran número de puntos en la superficie de un objeto y generan un fichero a partir de esos datos con una nube de puntos. La nube de puntos representa el conjunto de puntos en las tres dimensiones que ha medido el dispositivo y es generada a partir de la medición de ángulos y distancias mediante un haz de luz láser. La medición con estos dispositivos actualmente se basa sobre todo en cuatro principios mostrados en la Figura 2.2.

- **Triangulación**: Se lanza un haz láser que posteriormente será detectado por un sensor tras rebotar en la superficie de un objeto. El principio de medición está basado en el cálculo del triángulo formado por los componentes internos del escáner (el lugar desde donde se lanza el láser y el lugar donde es visto por el sensor) y el elemento a medir (la zona del objeto en la que rebota el láser). Sabiendo desde donde se lanza el haz y donde es leído por el sensor tras rebotar en el objeto se puede conocer la posición del punto del objeto en el que ha rebotado.

- **Tiempo de vuelo mediante pulsos**: Se lanza un haz de luz láser que alcanza al objeto y vuelve para ser detectado. De esta forma se determina la distancia en función del tiempo que tarda el haz en recorrer el doble de la distancia entre el emisor y el objeto.

- **Tiempo de vuelo mediante diferencia de fase**: En este caso el haz de luz láser emitido se propaga según ondas sinusoidales con una longitud de onda conocida. La distancia a medir se calcula en función del desfase entre la onda emitida y la reflejada en el objeto.

- **Luz estructurada**: Consiste en proyectar un patrón de luz en el objeto, generalmente una serie de líneas o una matriz de puntos, y analizar la deformación producida en este debido a la geometría de la escena. Esta se mide gracias a uno o más sensores (o cámaras), ligeramente desplazados del proyector, que observan la forma del patrón de luz y calculan la distancia de cada punto.

En el trabajo «*The 3D Model Acquisition Pipeline*»[10] se explica como obtener los datos usando estas técnicas y como producir con ellos un modelo 3D.

Dos de los sensores de bajo coste más utilizados para la adquisición de datos 3D son el sensor Microsoft Kinect[11] y el *Structure sensor*[1]), que se usa en este proyecto. En el caso del Microsoft Kinect se utiliza una tecnología de tiempo de vuelo mientras que el *Structure sensor* utiliza luz estructurada, proyectando con un láser un patrón de puntos en la zona que permite que después la cámara infrarroja pueda leer la profundidad.

Reconstrucción 3D a partir de múltiples imágenes. La segunda forma para obtener una nube de puntos, la fotogrametría digital, es aquella que utiliza múltiples imágenes de un objeto desde diferentes ángulos, para generar una representación 3D del objeto. Se obtienen medidas fiables de objetos físicos y su entorno a través de la medida e interpretación de imágenes. Con una sola foto se

[1] `https://structure.io/structure-sensor`

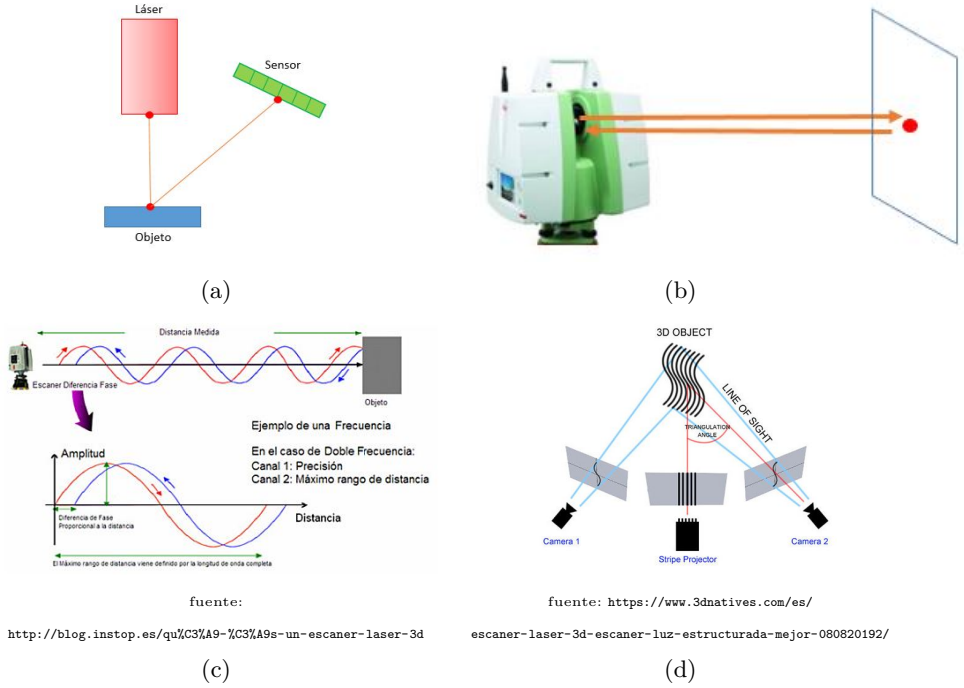


Figura 2.2: Diferentes técnicas para obtener nubes de puntos con escáneres láser 3D. (a) Triangulación. (b) Tiempo de vuelo mediante pulsos. (c) Tiempo de vuelo mediante diferencia de fase. (d) Luz estructurada.

puede obtener información bidimensional de la geometría del objeto. Si se trabaja con dos fotos de la misma escena capturadas desde dos ángulos diferentes, en la zona común a éstas (llamada zona de solape) se puede tener visión estereoscópica, es decir, información tridimensional. La fotogrametría añade más posiciones de cámara, rodeando el objeto a escanear con ellas, lo que la convierte en un digitalizador 3D más robusto. También se puede aplicar esta técnica a los fotogramas de un vídeo teniendo en cuenta que hay que elegir fotogramas grabados desde diferentes posiciones de la cámara, lo que hace este método menos consistente pero también más accesible y adaptable a diferentes aplicaciones.

En el libro «*Computer Vision: Algorithms and Applications*»[12] se explican de manera detallada estas técnicas de estimación de la posición o calibración externa a partir de imágenes en 2D y su aplicación en la realidad aumentada.

La fotogrametría es más económica que el uso de escáner láser 3D ya que no precisa de hardware adicional más allá de las cámaras, pero también es de menor calidad de captura 3D.

Las nubes de puntos tienen múltiples **aplicaciones**. Pueden usarse en la construcción tanto para planificar previamente y analizar los lugares donde ubicar la construcción como para verificar el progreso mientras se construye. Dentro de la industria puede utilizarse para realizar inspecciones de calidad y en metrología, para obtener medidas muy precisas. También sirve para comprobar el estado de edificios y lugares históricos, o simplemente para tener un modelo de

ellos que permita su visita virtual y el turismo visual. A pesar de su multitud de utilidades, en muchas aplicaciones tridimensionales no se usan las nubes de puntos de esta forma. Se convierten en modelos de mallas, de los que se habla en la siguiente sección 2.2, o modelos de CAD mediante un proceso de reconstrucción de superficies.

Los **modelos de nubes de puntos usados en este trabajo** son modelos de corales reales encontrados en el fondo marino para recrear un escenario submarino. Estos datos han sido proporcionados por el *Marine Imaging Lab*[2], grupo de investigación colaborador del laboratorio de investigación en el que se realiza este trabajo. Los modelos fueron tomados gracias a una cámara submarina y construidos y mejorados usando Matlab[3] y Agisoft[4]. Muchos de estos modelos se encuentran en Sketchfab[5] y en la Figura 2.3 se puede ver una muestra de alguno de ellos.

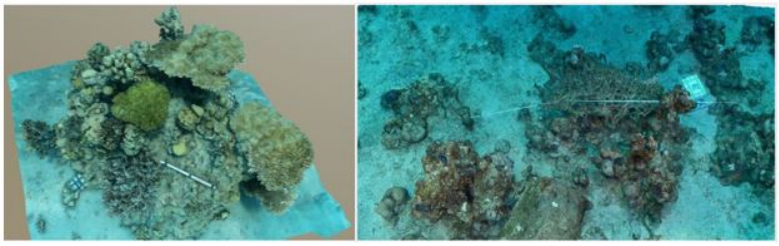


Figura 2.3: Modelos de corales de nubes de puntos utilizados para este proyecto.

2.2. Modelos de mallas 3D

Las mallas 3D están constituidas por pequeños planos poligonales, normalmente triángulos, formando un objeto cerrado. Estos modelos se pueden obtener de nubes de puntos, de manera que la malla se adapta a la forma real del objeto modelado mediante la nube de puntos. Cada vértice de los triángulos es uno de los puntos tridimensionales y cada arista es la unión entre dos vértices. De esta forma se crean los triángulos que forman las superficies que unidas constituyen la malla. Esto viene explicado en el trabajo «Segmentación y clasificación de mallas 3D»[13]. Las mallas también pueden ser creadas directamente en *softwares* para gráficos 3D como pueden ser programas CAD o basados en modelos matemáticos, pero esto son modelos creados en un programa no capturados de un entorno real.

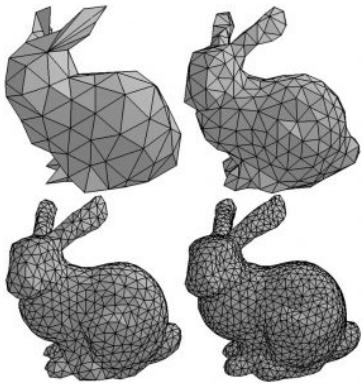
Tipos de mallas. Aunque lo habitual es que las mallas poligonales estén formadas por triángulos, ya que las tarjetas gráficas están diseñadas para acelerar los cálculos con ellos e internamente trabajan con estas superficies convirtiendo otras representaciones en mallas trianguladas, hay más representaciones de mallas 3D entre las que se pueden destacar:

[2] `http://csms.haifa.ac.il/profiles/tTreibitz/index.html`
[3] `https://www.mathworks.com/products/matlab.html`
[4] `https://www.agisoft.com/`
[5] `https://sketchfab.com/Marine_Imaging_Lab`

- **Mallas poligonales**: Estas están formadas por polígonos que pueden ser de 4 lados, llamados Quads, o de N lados, a veces llamados Ngons. Estas son generalmente más cómodas de manipular por parte del diseñador pero, como se ha comentado anteriormente, internamente están formadas por triángulos.
- **Mallas subdivision (Sub-D)**: Estas consisten en una malla, habitualmente poligonal, a la que se le ha aplicado un algoritmo de subdivisión que añade puntos adicionales intermedios y que según el algoritmo mueve la posición de algunos vértices para dar suavidad a la malla. El proceso de subdivisión puede repetirse varias veces hasta obtener el resultado esperado. Este tipo de mallas es adecuado para el modelado de formas geométricas complejas y con muchas curvas, como el modelado orgánico, a la vez que también es adecuado con modelos con muchas caras planas y aristas vivas.

Los tipos de mallas 3D vienen detallados en el libro «Geometric Modeling Based on Polygonal Meshes»[14], donde además se explican más detalles de la representación 3D de superficies como su calidad o el suavizado de las mismas.

Las mallas 3D pueden tener distintas resoluciones en función del tamaño de los triángulos que la forman, como se puede observar en la Figura 2.4. Dependiendo del uso que se le va a dar al modelo se elegirá una resolución mayor o menor, aunque esta también dependerá de las limitaciones del *hardware* con el que se capture el modelo. Por ejemplo, si se crea un modelo con el objetivo de imprimirlo este tiene que tener una consistencia y solidez mayor que si este modelo se va a utilizar en un videojuego o en alguna aplicación virtual. Esto se debe a que para imprimirlo su superficie ha de ser continua mientras que para visualizarlo en un videojuego o aplicación se busca un buen aspecto visual y que parezca lo más real posible a la vez que la rapidez y facilidad de su representación en la pantalla.



fuente: `https://disenoimpresion3d.es/tutoriales-impresion-en-3d/que-es-la-impresion-en-3d-2/obtencion-de-modelos-3d/`

Figura 2.4: Mallas 3D de diferentes resoluciones.

Modelos utilizados en este trabajo. En este proyecto, los modelos de mallas 3D se han tomado con un *structure sensor*, mostrado en la Figura 2.5a, del que se explica el funcionamiento en el apartado sensores de profundidad de la sección anterior 2.1, y un iPad, utilizando aplicaciones existentes para capturar tanto objetos como habitaciones y entornos de interior, como se puede ver en los ejemplos de la Figura 2.5b. El software utilizado sirve tanto para la captura los datos, como para la construcción de la malla 3D.

(a) (b)

Figura 2.5: Structure sensor y modelos capturados por él. (a) Structure sensor utilizado montado en el iPad. (b) Modelos 3D tomados con el structure sensor. A la izquierda modelo de una hucha tomado con la App Scanner para iOS. A la derecha malla 3D de una espacio de interior mientras se va capturando con la App Room Capture para iOS.

2.3. Técnicas de importación de los datos

Esta sección describe como exportar los modelos anteriormente explicados al ordenador e importarlos al motor gráfico.

En el caso de los modelos capturados con el *structure sensor*, desde sus aplicaciones para iOS (como *Room Capture*[6] para escenarios de interior y *Scanner*[7] para objetos), se pueden previsualizar los modelos directamente en el iPad. Desde dicha visualización existe una opción de mandar directamente al correo electrónico el modelo escaneado, de manera que se envíe un archivo comprimido (ZIP) con el modelo 3D en formato .obj y la textura en formato .jpg, además de una imagen con la previsualización del modelo fuera de este archivo comprimido. En el caso del escaneo de objetos, también se permite exportarlos a CAD y a otros software para impresión 3D, pero para el uso de los modelos que se pretende en este proyecto la opción más adecuada y sencilla es enviar el archivo por correo.

En el caso de los modelos de nubes de puntos utilizados en la aplicación del fondo submarino utilizados de Sketchfab, se obtiene directamente el fichero descargándolo desde esta página. Lo único que se ha de hacer es crearse una cuenta en dicha página y buscar el modelo que se quiera. Al pulsar en ese modelo se abre una previsualización y debajo se puede dar a la opción *Download 3D model*, en la que si se hace *click* aparece el formato en el que está el modelo y el botón para

[6] `https://apps.apple.com/us/app/room-capture-structure-sdk/id916501905`
[7] `https://apps.apple.com/us/app/scanner-structure-sdk/id891169722`

descargarlo. No todos los modelos de corales utilizados están en Sketchfab, algunos de ellos han sido proporcionados directamente por el *Marine Imaging Lab*.

Una vez se tienen el modelo 3D y la textura de este, se procede a su importación al motor gráfico *Unreal Engine*. Este procedimiento es sencillo, ya que solo hay que pulsar en el botón importar que se puede ver en la Figura 2.6.

Figura 2.6: Botón para importar modelos en Unreal Engine 4. En rojo la ventana del navegador de contenido ("*Content Browser*") dentro de la que está el botón de importar ("*Import*") en amarillo.

A la hora de importar lo que hay que tener en cuenta es que el modelo esté en uno de los formatos que admite el motor, al igual que el archivo de la textura. Esto no tiene por que ser un problema, ya que algunos de los formatos más comunes de modelos como el .obj o el .fbx y de texturas como .jpeg, .jpg o .png son admitidos en el programa. Sin embargo, es común encontrar modelos en archivos .stl debido a que estos son muy usados en las impresoras 3D y dentro de Unreal no se puede importar este tipo de archivos. La solución a esto es utilizar algún otro programa donde se puedan importar, como puede ser Blender[8], para después exportarlo en otro formato compatible con Unreal como por ejemplo .fbx. De la misma manera, se pueden transformar otros tipos de archivos para hacerlos compatibles. Todos los formatos de archivo que admite Unreal son: *.3g2, *.3gp, *.3gpp, *.3gp2, *.ac3, *.amr, *.abc, *.m4a, *.m4v, *.mov, *.asf, *.upack, *.as, *.adts, *.au, *.aif, *.aiff, *.avi, *.bwf, *.csv, *.cdda, *.caf, *.hdr, *.exr, *.fbx, *.fga, *.ies, *.json, *.aac, *.mp3, *.mp4, *.WAV, *.obj, *.otc, *.otf, *.sdv, *.snd, *.srt, *.st, *.paper2dsprites, *.sami, *.smi, *.bmp, *.float, *.jpeg, *.jpg, *.pcx, *.png, *.psd, *.tga, *.dds, *.ttc, *.ttf, *.udn, *.wave, *.webm, *.wma, *.wmv.

También se ha de tener en cuenta que hay un límite máximo de tamaño de 8192x8192 píxeles en los archivos de textura. Por lo tanto si se tienen texturas de alta resolución que superen este tamaño, Unreal no va a ser capaz de importarlas. La solución es simplemente reducir el tamaño del archivo por debajo de ese máximo para poder importarlo sin ningún problema. Esto para los formatos de archivo más comunes como son .jpeg o .jpg se puede hacer con multitud de programas, algunos tan sencillos como Paint [15].

[8] `https://www.blender.org/`

Capítulo 3

Sistema para realidad virtual/aumentada

Este capítulo describe los componentes principales utilizados para diseñar y construir los demostradores de realidad virtual utilizando modelos 3D de escenarios reales (descritos en el capítulo anterior 2). Lo principal en cuanto a hardware es el *headset* de realidad virtual. Por otro lado, en cuanto a software se usan Unreal Engine 4, el software específico del *headset* utilizado, SteamVR, y algunos programas de procesado de modelos 3D en menor medida. En la Figura 3.1 se muestran los componentes nombrados.

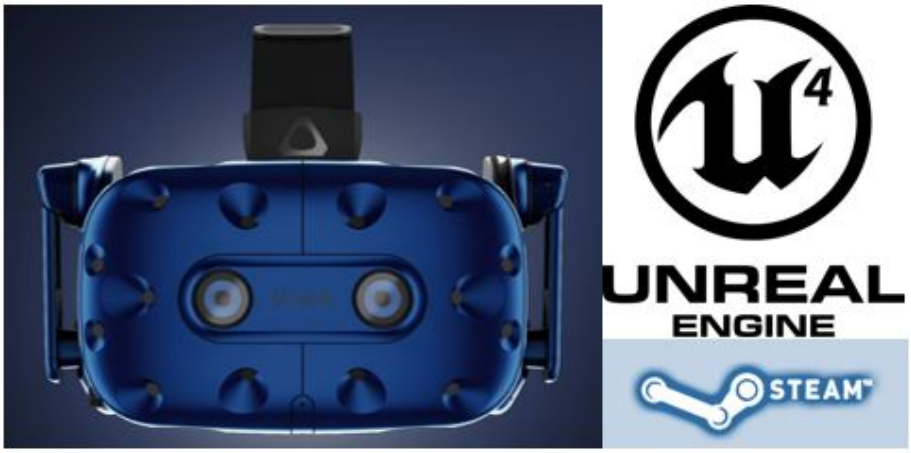


Figura 3.1: Componentes principales del sistema, tanto hardware como los logos de los softwares utilizados.

3.1. Hardware

3.1.1. Gafas de realidad virtual

Alternativas en el mercado. Actualmente hay multitud de gafas de realidad virtual en el mercado, de las cuales algunas se han nombrado en la introducción en la sección 1.1. Existen modelos móviles, en los que el vídeo se reproduce desde un *smartphone*, y modelos estacionarios, en los que es necesario que un ordenador mande el vídeo a través de un cable (ya sea HDMI o cable DisplayPort en los modelos más actuales). Dos conocidos ejemplos de gafas de realidad virtual móviles son las Google Cardboard VR y las Samsung Gear VR, que aparecen en la Figura 3.2.

- **Google Cardboard VR.** Estas están formadas por dos lentes y un cartón plegable, en el que se encaja el *smatphone*, que es el encargado de determinar si se está moviendo la cabeza gracias a su acelerómetro y giroscopio. Debido a los materiales que usa y a que no necesita un potente ordenador y un software específico, es la opción más asequible del mercado pero también la más simple.
- **Samsung Gear VR.** Estas tienen más controles que las anteriores, por ejemplo una ruleta para ajustar la alineación de nuestros ojos, otra para el volumen o una superficie táctil. También se puede comprar un mando para facilitar el control e interacción con los entornos virtuales. Además incorporan una IMU (*Inertial Measurement Unit*). Este dispositivo es un conjunto de acelerómetros y giroscopios que controla hacia donde se mueve la cabeza y con que velocidad de manera más precisa que solo con el *smartphone*. Tienen un ángulo de visión de 101º y gran variedad de aplicaciones, experiencias y juegos que se pueden probar.

(a) (b)

Figura 3.2: Dos modelos de gafas de realidad virtual móviles. (a) Google Cardboard VR. (b) Samsung Gear VR.

Dentro del otro tipo de gafas de realidad virtual, los equipos estacionarios, se encuentran las HTC Vive Pro utilizadas en este proyecto. Además de estas hay otras alternativas como por ejemplo Oculus Rift y Play Station VR, las cuales se muestran en la Figura 3.3.

- **Oculus Rift.** Estas gafas crean un efecto estereoscópico en 3D a partir de unas lentes especiales situadas entre las pantallas y los ojos, las cuales redefinen las imágenes del vídeo para cada ojo. Para seguir el movimiento y determinar la posición del visor utilizan una combinación de giroscopios

de 3 ejes, acelerómetros y magnetómetros (IMU). Proporcionan 110º de visión, los *displays* se refrescan a 90 fps (*frames per second*) y tienen un retardo entre el movimiento de la cabeza y el de las imágenes que se muestran de 30ms.

- **Play Station VR.**[1] Funcionan de la misma manera que las Oculus Rift, añadiendo además el uso de 9 LEDs que tiene distribuidos por el HMD (*head-mounted display*) visibles por una cámara conectada a Play Station 4 que hace que el posicionamiento del visor sea más preciso. En este caso, los *displays* se refrescan a 120 fps y el retardo entre el movimiento de la cabeza y de las imágenes mostradas es de tan solo 18ms.

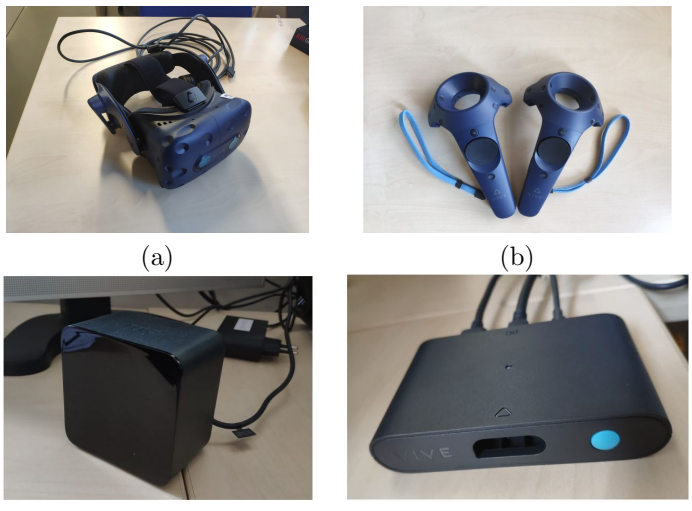
(a) (b)

Figura 3.3: Dos modelos de gafas de realidad virtual conectados por cable. (a) Oculus Rift con su mando. (b) Play Station VR.

HTC Vive Pro. Este *headset* es el elegido para este proyecto, ya que además de ser uno de los más nuevos (lanzamiento en 2018), utiliza un sistema de seguimiento llamado *Lighthouse* que permite que te puedas levantar y andar dentro de un espacio sin que ni el visor ni los controladores pierdan su posición y orientación gracias a que el sistema posee 6 GDL (grados de libertad), 3 de orientación y 3 de posición. Esto las diferencia de otras gafas existentes, en las que el seguimiento está pensado para experiencias estáticas. Los dispositivos de realidad virtual/aumentada de bajo coste (Google Cardboard VR, Samsung Gear VR) solo poseen los 3 GDL de rotación, también otros sistemas más sofisticados como Oculus Rift y Play Station VR tienen estos 3 GDL y pueden tener datos de posición si se les añade una cámara externa para el seguimiento.

El *set* completo Vive Pro se compone del visor, dos controladores y dos estaciones base que son imprescindibles para que un seguimiento preciso tanto de las gafas como de los controladores sea posible. Además de los **componentes** principales mostrados en la Figura 3.4, el *set* viene equipado con todos los cables y accesorios necesarios para su uso y mantenimiento. Las estaciones base traen su alimentación y un kit de montaje para fijarlas en la pared y los controladores también vienen con sus cables correspondientes para cargarlos. Además otra parte importante del *set* es la caja *LinkBox*, que permite la conexión de las gafas al ordenador de manera sencilla. Todos los componentes con sus sensores y accesorios se encuentran detallados en el apéndice A, además de las especificaciones del *headset*.

[1] `https://www.playstation.com/es-es/explore/playstation-vr/`



(a) (b)

(c) (d)

Figura 3.4: Componentes principales del *headset* completo HTC Vive Pro. (a) Gafas de realidad virtual. (b) Controladores. (c) Estación Base. (d) *Linkbox*

El sistema de seguimiento y posición ***Lighthouse* o Faro** está desarrollado por Valve Software[2] una empresa desarrolladora de videojuegos, de hardware y creadora de Steam[3], una plataforma de distribución digital de videojuegos. Este sistema se encuentra tanto en el Vive Pro HMD como en sus controladores. Su funcionamiento viene explicado en el capítulo 4 del libro «*Understanding Virtual Reality: Interface, Application, and Design*»[16] al igual que otras tecnologías de seguimiento usadas en la realidad virtual. El sistema se basa en lo siguiente. Las dos estaciones base están compuestas por un grupo de LED estáticos y dos emisores láser que giran a alta velocidad. Los LED destellan 60 veces por segundo a la vez que uno de los emisores láser realiza un barrido (alternando entre uno horizontal y otro vertical). Las dos estaciones están sincronizadas de forma que realizan este proceso a la vez. Tanto el visor como los controladores están cubiertos de sensores que detectan los destellos LED y el láser, de manera que en cuanto se localiza un destello el visor empieza a cronometrar el tiempo que tarda el láser en alcanzar alguno de sus sensores, calculando con ello después la posición del visor/controladores relativa a las estaciones base. Esta técnica combinada con IMU's permite un seguimiento del movimiento muy preciso. Más detalles de este sistema se explican en el apéndice A.

Una vez conocido su funcionamiento, hay que estudiar su **conexión** al ordenador para poder empezar a utilizarlo. Esta conexión viene explicada de forma breve en la propia página de Vive[4] y de forma más detallada en su guía «*VIVE Pro HMD User guide*»[3], donde además vienen descritos cada uno de los accesorios del *set*, su funcionamiento, preguntas frecuentes y otros ajustes. Gracias a esta información la tarea de conectar las gafas y ponerlas en marcha se facilita. Esto viene explicado paso a paso en el apéndice A .

[2] `https://www.valvesoftware.com/es/`
[3] `https://store.steampowered.com/?l=spanish`
[4] `https://www.vive.com/us/setup/vive-pro-hmd/`

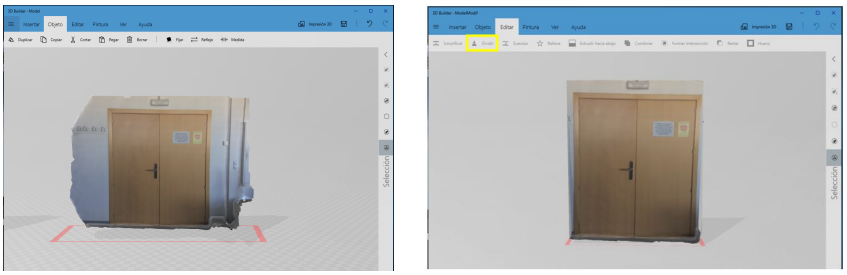
3.2. Software

3.2.1. Procesado de Modelos 3D

Antes de importar los modelos al motor *Unreal Engine*, se han necesitado hacer pequeñas modificaciones en algunos de ellos.

Paint. Se ha utilizado la herramienta Paint para reducir el tamaño de las texturas con una resolución mayor que la que se permite importar en Unreal, como se explica en la sección 2.3.

3D Builder. Se ha utilizado esta herramienta de Microsoft, 3D Builder[5], para recortar uno de los modelos tomados con el *structure sensor* y así evitar que tuviera un contorno tan irregular para después poder integrarlo más fácilmente en el escenario de Unreal. Primero se ha creado una escena nueva en la que se ha introducido el modelo desplegando el menú Insertar → Agregar → Cargar Objeto. Después se ha utilizado la herramienta Dividir que se encuentra en el menú Editar. En la Figura 3.5 se puede ver el cambio en el modelo tras realizar este proceso.



(a) (b)

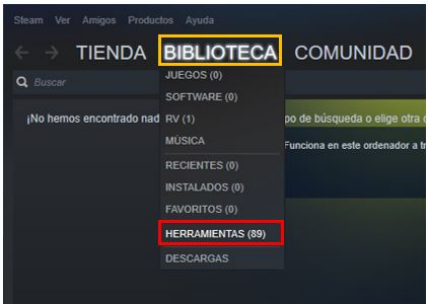
Figura 3.5: Antes y después del modelo editado en 3D Builder. (a) Modelo importado al programa 3D Builder antes de ser modificado. (b) Modelo tras ser editado con la herramienta Dividir que aparece recuadrada en amarillo.

3.2.2. SteamVR

Para que las gafas de realidad virtual Vive Pro funcionen y se puedan conectar al ordenador y que después Unreal trabaje con SteamVR, se debe descargar un software específico. Es necesario instalar Steam en el ordenador, descargándolo desde su página web[6], para después poder acceder a SteamVR, el ecosistema de realidad virtual que utilizan las Vive Pro. Aunque es gratis hay que crear una cuenta con la que, una vez iniciada la sesión, se podrá lanzar SteamVR buscando dentro del software Steam en la pestaña Biblioteca → Herramientas la opción SteamVR[beta] y haciendo *click* en ella, como se muestra en la Figura 3.6.

[5] `https://www.microsoft.com/es-es/p/3d-builder/9wzdncrfj3t6?activetab=pivot:overviewtab`

[6] `https://store.steampowered.com/`



(a)

(b)

Figura 3.6: Ventanas que aparecen al iniciar Steam donde encontramos la herramienta SteamVR. (a) Steam con el menú Biblioteca desplegado en color amarillo y en color rojo la opción Herramientas. (b) Herramienta SteamVR[beta] que debe ejecutarse.

También antes de empezar a utilizar el *headset* se puede escalar la experiencia de realidad virtual definiendo el área de juego del que se dispone. Desde SteamVR se pulsa la flecha para desplegar las opciones y se selecciona *Run Room Setup*. Una vez pulsado esto solo se han de seguir las instrucciones que aparecen.

Después de esto y conectar todos los componentes del *headset* ya se puede ver si el equipo está preparado y bien conectado para ser utilizado, como se ve en la Figura 3.7. Además una vez en uso, SteamVR lanza un aviso si se está demasiado cerca de los límites del área de juego definida en el mundo real, gracias a una tecnología llamada *Chaperone*.

Tras todos estos ajustes iniciales, ya se pueden visualizar los proyectos que se realicen en Unreal en las gafas de realidad virtual e interactuar con ellos. No obstante, que los proyectos realizados en Unreal Engine puedan verse en las gafas de realidad virtual, SteamVR debe permanecer abierto y ejecutándose.

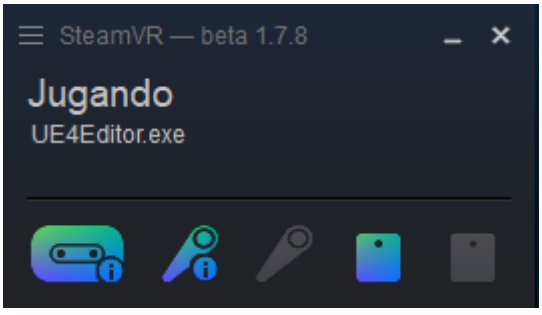


Figura 3.7: Ventana de SteamVR que muestra el visor, un controlador y una estación base conectados.

3.2.3. Unreal Engine

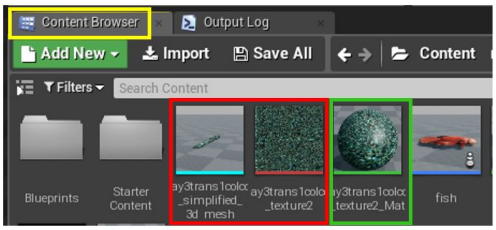
Alternativas. Para desarrollar una aplicación de realidad virtual es necesario un motor gráfico donde crear el escenario que posteriormente se visualizará en el HMD. Actualmente existen varios motores entre los que puedes elegir dependiendo de las características y el posterior uso de las aplicaciones desarrolladas en ellos que se deseen. Entre estos los más utilizados en el ámbito del desarrollo de videojuegos e implementación de aplicaciones para realidad virtual son Unity[7] y Unreal Engine[8].

- **Unity.** Está desarrollado por la compañía *Unity Technologies* y es uno de los motores gráficos más utilizados ya que tiene una gran versatilidad. Permite crear juegos (ya sean para móviles, para consolas, para PC, de realidad aumentada o de realidad virtual), películas, animación, cinemáticas y experiencias inmersivas en 3D. Dispone de tres opciones: la personal para principiantes que es gratuita pero no incluye algunas características, la Plus para aficionados y la Pro para equipos y profesionales. Estas dos últimas son versiones de pago. A pesar de que es un software ampliamente conocido y utilizado, este proyecto se ha decantado por utilizar Unreal Engine con el que actualmente están desarrollados videojuegos conocidos y con buena calidad y que además se puede obtener gratis con todas sus herramientas.

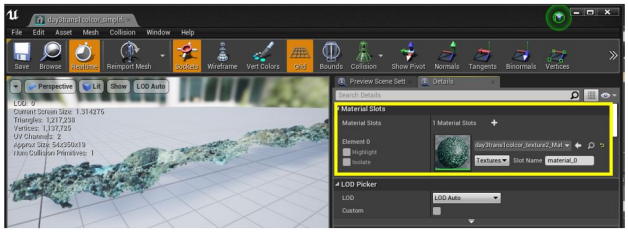
- **Unreal Engine.** Es el motor gráfico desarrollado por Epic Games[9] que se ha decidido utilizar en el proyecto ya que es de los más utilizados actualmente y cuenta con gran cantidad de documentación y soporte. Sirve para desarrollar videojuegos y aplicaciones para PC, consolas, móviles, realidad virtual y realidad virtual, programando en C++ y en Blueprints, que se basa en el uso de una *interface* gráfica basada en nodos. Estos hacen que la programación sea más sencilla ya que es más visual y hay muchos bloques con funciones ya implementadas. Este programa cuenta con todas las herramientas necesarias tanto para empezar como para profesionales y se puede descargar gratis desde su página. En este trabajo se ha utilizado la versión del software Unreal Engine 4.21.

Carga de modelos 3D. Para cargar los modelos dentro de Unreal primero se han de importar como se explica en la sección 2.3. Después de esto, el modelo importado y su textura aparecen en la ventana de *Content Browser*. Para colocar la malla del modelo dentro del escenario solo hay que hacer *click* en ella y arrastrarla hasta la escena del nivel. Para asignarle la textura al modelo se arrastra y se suelta encima de la malla que se acaba de colocar, tras lo que se creará automáticamente un material que se llama *NombreTexturaImportada_Mat*. Luego se hace doble *click* en la malla estática del modelo que aparece en el apartado de *Content Browser* y se abrirá una ventana de un editor en la que se debe pulsar en *Details* → *Material Slots*, buscar ahí el nombre del material que se acaba de crear y cambiarlo en caso de que no se haya cambiado ya de manera automática. En la Figura 3.8 se ven las ventanas y pasos a seguir para realizar este proceso.

[7] `https://unity.com/es`
[8] `https://www.unrealengine.com/en-US/`
[9] `https://www.epicgames.com/store/es-ES/`



(a)



(b)

Figura 3.8: Ventanas y pasos a seguir para cargar un modelo en Unreal Engine. (a) Pestaña *Content Browser* en amarillo, donde aparecen la malla del modelo y su textura en rojo tras ser importados y en verde el material que se crea al colocar la textura encima del modelo en el escenario. (b) Ventana de la malla estática del modelo con la opción *Material Slots* para cambiar el material en amarillo.

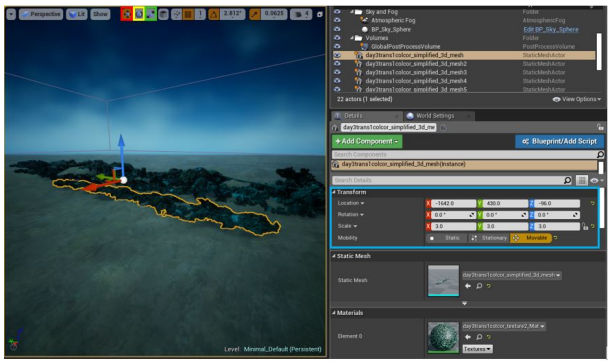


Figura 3.9: Arriba herramientas para trasladar (en rojo), para rotar (en amarillo) y para escalar (en verde) el modelo. A la derecha en azul variables que se pueden modificar para conseguir lo mismo.

Por último una vez que el modelo está ya en el escenario con su textura, este se puede modificar. Se puede cambiar su posición, su orientación y su tamaño. Esto se puede hacer de dos maneras: pulsando en el modelo y utilizando los controles que aparecen arriba a la derecha en la ventana del nivel o una vez seleccionado el modelo, en la pestaña *Details* → *Transform*, aparecen las variables X Y y Z tanto de posición como de orientación como de escala que se pueden cambiar desde ahí. Estas dos formas de mover y cambiar de tamaño el modelo dentro del escenario se ven reflejadas en la Figura 3.9.

Creación del entorno

- **Entorno natural submarino.** En Unreal, por defecto al crear un nuevo proyecto, el fondo es un cielo azul con nubes y hay un pequeño trozo de suelo artificial. Si se quiere recrear un ambiente nocturno, con niebla, con un suelo de tierra, con hierba, con alguna variación en el terreno o cualquier fondo que se nos ocurra para un escenario que imite un entorno natural, se debe modificar tanto el cielo como el terreno.

  Para modificar el cielo basta con desplegar la carpeta *Sky and Fog* que aparece en la ventana *World Outliner* y cambiar las opciones de *Atmospheric Fog* y de *BP_Sky_Sphere* que se deseen. Estas opciones se pueden ver en la Figura 3.10. En este trabajo las únicas variaciones que se han realizado son en *BP_Sky_Sphere*, ya que lo único que se necesitaba era eliminar las nubes para que no se vieran de fondo en la escena marina.

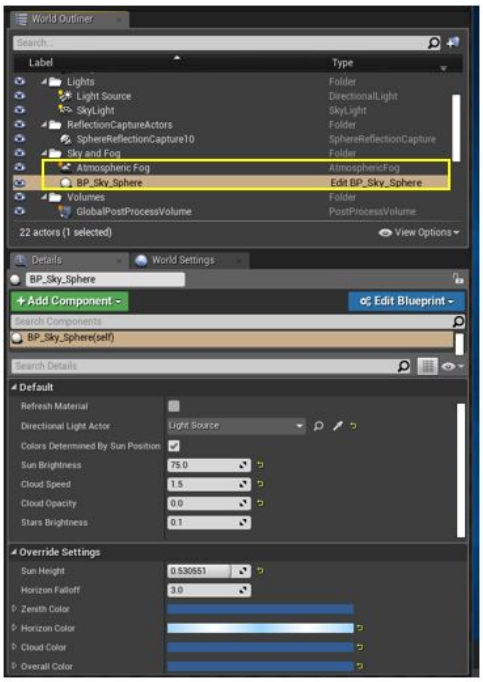


Figura 3.10: En amarillo los dos elementos de la escena que se pueden variar para modificar el cielo y el ambiente en ella. Debajo las opciones de *BP_Sky_Sphere* que se han cambiado en este proyecto.

En el caso de la creación de un terreno natural, se puede hacer entrando en las pestañas *Modes* → *Landscape*. Una vez aquí marcando la opción *Sculpt*, se pueden formar desniveles en el terreno. Esto es lo que se ha utilizado para crear hoyos y montículos que imiten la arena del fondo submarino. Para finalizar, se puede cambiar el *Landscape Material*. En este caso se ha elegido uno que parezca arena, pero hay mucha variedad para elegir pudiendo escoger por ejemplo uno que recree la hierba o la roca. Todo esto se puede ver en la Figura 3.11.

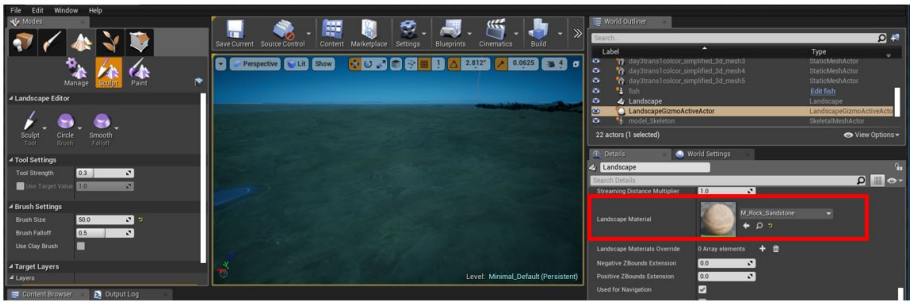


Figura 3.11: A la izquierda aparece seleccionado el modo *Sculpt*. En rojo, a la derecha, el *Landscape Material* elegido. En el centro, en el escenario, una muestra de cómo queda el terreno tras las modificaciones.

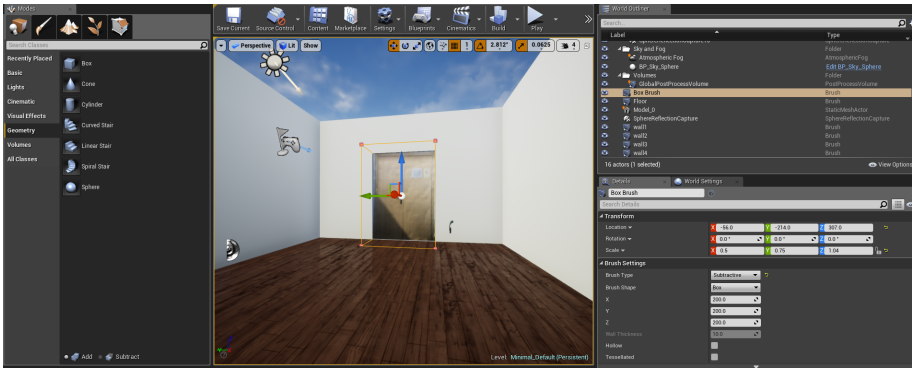


Figura 3.12: A la izquierda aparece seleccionado la pestaña *Geometry* donde se encuentran las cajas. Abajo a la derecha, la pestaña donde se elige el *Brush Type*. En el centro, en el escenario, la forma básica de la habitación que se recrea.

- **Escenario de interior.** Si lo que se desea es crear un escenario de interior como puede ser una habitación o pasillo, lo que hay que hacer es construir tanto las paredes como el suelo. Para ello desde *Modes* → *Place* → *Geometry*, se añaden varias cajas, una para modelar el suelo y una para cada pared. Desde aquí también se pueden añadir escaleras si son necesarias en el escenario que se está creando. Una vez se tienen estas cajas se escalan para crear paredes del tamaño y grosor que se desee y se colocan alineándolas de manera precisa desde la vista *Top*. Después se procede a la creación de huecos para las ventanas y puertas, creando otra caja, colocándola donde se quiera que esté el hueco y una vez ahí cambiando en *Deatails* → *Brush Settings* la opción *Brush Type* de *additive* a *Subtractive* para que en lugar de un bloque macizo sea un hueco. Tras esto se pueden insertar modelos de puertas y ventanas y encajar en esos huecos. Finalmente, se puede cambiar el *Surface Material* a las superficies creadas, eligiendo el que más se adecúe en cada caso. Estas opciones y el resultado tras aplicarlas se muestran en la Figura 3.12.

En ambos casos, estas indicaciones son solo para crear un escenario básico. Después de llevar a cabo estos pasos se pueden seguir incluyendo modelos (ya

sean creados, tomados de páginas como Sketchfab[10] o capturados en entornos reales) para hacer el escenario más realista.

Efecto "agua". Por defecto, al crear un nuevo proyecto en Unreal el fondo de la escena que aparece es un cielo con nubes. Para crear un ambiente en el escenario que simule las condiciones que hay debajo del agua y crear un efecto más realista, hay que utilizar un *post process volume* buscándolo en el *Search Classes* y arrastrándolo a la escena. Dentro de él será la zona en la que se consiga el efecto "agua", por lo que hay que escalarlo hasta lograr el tamaño suficiente para que cubra el espacio en el que se desee este efecto. Las opciones que hay que modificar se encuentran, una vez seleccionado el *post process volume*, en la ventana *Details*. En *Details* → *Lens* → *Depth of Field* se cambian las opciones *Method*, *Focal Distance* (distancia en la que el efecto es nítido), *Focal Region* (región en la que el contenido está enfocado, empezando después de *Focal Distance*), *Near Transition Region* y *Far Transition Region* (definen la region de transición que está al lado de la *Focal Region*). Todas estas opciones aparecen en la Figura 3.13.

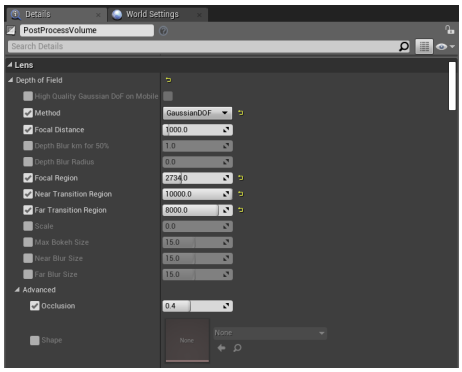


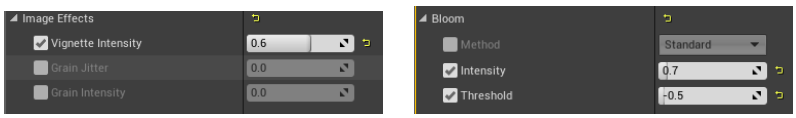
Figura 3.13: Opciones del *post process volume* a seleccionar en la pestaña *Depth of Field*.

En *Details* → *Lens* → *Image Effects* se modifica *Vignette Intensity*. En *Details* → *Lens* → *Bloom* se varían *Intensity* y *Threshold* para los brillos, probando hasta obtener el resultado deseado. Los cambios realizados en *Image Effects* y en *Bloom* se muestran en la Figura 3.14.

Por último, en *Details* → *Color Grading* → *Misc*, se modifica el *Scene Color Tint* para lograr ver el fondo azulado. El panel de opciones para modificar el color de la escena con los valores que se han elegido aparece en la Figura 3.15.

Además de permitir crear una ambientación de fondo marino de esta manera, Unreal también tiene la opción de crear un efecto que simule la superficie del agua. Esto se hace de manera sencilla, basta con elegir el material *M_Water_Ocean* o *M_Water_Lake*, según el propósito, y aplicarselo a la superficie en la que se precise ese efecto. Esta herramienta se ha descartado para este trabajo, ya que el escenario es en todo momento submarino.

[10] `https://sketchfab.com/`



(a) (b)

Figura 3.14: Opciones del *post process volume* para conseguir el efecto agua. (a) Opciones en el apartado *Image Effects*. (b) Opciones en el apartado *Bloom*.

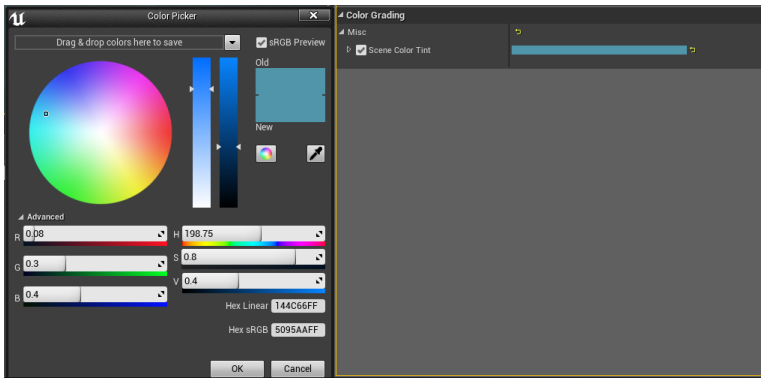


Figura 3.15: Opciones del *post process volume* a seleccionar en la pestaña *Scene Color Tint* con los valores elegidos.

Interacciones físicas. Para conseguir simular las interacciones físicas y las colisiones entre elementos, tan solo se tienen que marcar las opciones de la malla estática que se deseen recrear. Seleccionando el modelo en *Details* → *Physics* se encuentran estas opciones, como por ejemplo *Simulate Physics* que ha de estar marcada para que otros actores de la escena no atraviesen los elementos fijos o *Enable Gravity* con la que se puede simular la gravedad si está marcada o no hacerlo si no lo está.

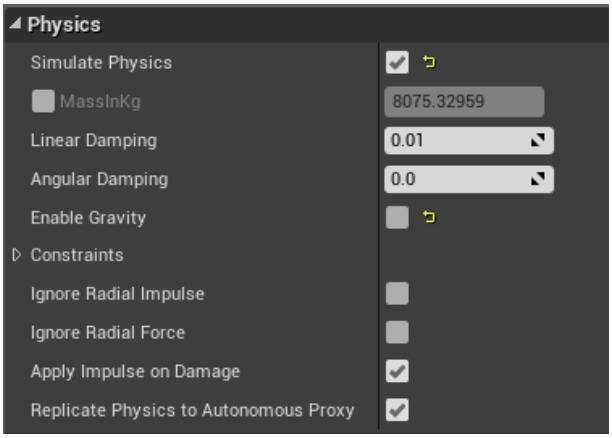


Figura 3.16: Algunas de las opciones que se pueden modificar para simular las interacciones físicas.

Trayectorias predefinidas. En ocasiones, se quiere que un actor de la escena se mueva siguiendo una trayectoria determinada sin necesidad de ser controlado por nadie. Esto puede darse, por ejemplo, en la reconstrucción de un escenario recreado que se quiera solamente mostrar y en el que haya fauna o en las animaciones de las cinemáticas de los videojuegos. Para conseguir que un personaje dentro de un escenario se comporte así es necesario crear un componente llamado *Spline*. Para ello se abre el editor del personaje que se quiera animar y en la pestaña *Viewport* se añade un nuevo componente *Spline*. Tras esto se va a la pestaña *Event Graph* y aquí se programa el seguimiento de la trayectoria con un bloque *Timeline*. Como esta componente es la mas compleja del programa de entre las estudiadas, el proceso para obtenerla se describe en mas detalle en el apéndice B. Una vez programado se cierra el editor del personaje y se va a la ventana principal del escenario donde se puede modificar la trayectoria que se seguirá. Haciendo *click* con el botón derecho se pueden añadir puntos a la ruta y marcar la opción *Visualize Roll and Scale*. Con esto se puede mover, rotar y alargar la trayectoria hasta obtener el resultado deseado. Esta trayectoria solo se realiza una vez, para lograr que el personaje se mueva repitiendo esta trayectoria continuamente se crea la ruta de manera que forme una trayectoria cerrada y se añade un bucle *while*. En la Figura 3.17 se ve el resultado tras crear el componente *Spline* para que un pez del escenario submarino siga una trayectoria cíclica.

Figura 3.17: Trayectoria creada vista tras marcar la opción *Visualize Roll and Scale* donde se ven los puntos creados en ella que permiten modificar la ruta que seguirá el personaje.

Entradas desde periféricos. Para poder interactuar con alguno de los personajes del escenario, controlándolos y moviéndolos por donde se desee, se deben configurar y añadir entradas desde el periférico que se quiera utilizar, ya sea teclado, ratón o controladores de RV. Esto se realiza pulsando la pestaña *Edit* → *Project Settings* → *Input* y aquí se añaden entradas del tipo *Action Mappings* o *Axis Mappings*. Las primeras sirven para llevar a cabo acciones puntuales como saltar, mientras que las segundas se pueden programar para acciones como mover a la izquierda/derecha o mover delante/detrás. La ventana de modificación

de las entradas se muestra en la Figura 3.18. Una vez realizado esto se programa, en la ventana *Event Graph* del personaje con el que se quiera interactuar, el movimiento que se desee realizar tras cada entrada, sin olvidar habilitarlas primero añadiendo un bloque *Enable Input* detrás del bloque del evento de cada entrada creada. Se ha de tener en cuenta también que si se quiere utilizar los controladores de realidad virtual se debe habilitar el *plugin* SteamVR en la pestaña *Edit* → *Plugins* → *Virtual Reality* para que funcione.

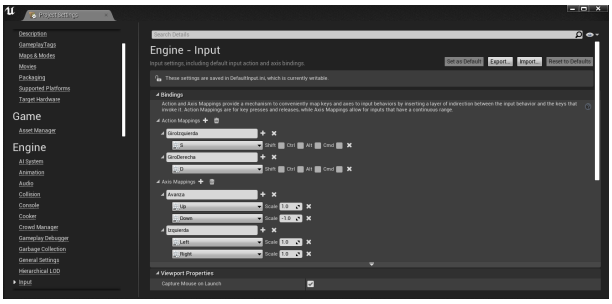


Figura 3.18: Ventana de creación y modificación de entradas en Unreal Engine.

Capítulo 4

Demostradores desarrollados

Este capítulo describe la utilidad y el propósito de cada una de las dos aplicaciones desarrolladas. Muestra también los elementos que se han integrado en cada una de ellas y las herramientas del *software* explicadas en el apartado Unreal Engine de la sección 3.2 utilizadas para ello. También incluye un pequeño manual con las instrucciones a seguir para poder ver en el visor de realidad virtual los demostradores diseñados.

4.1. Experiencia inmersiva en un escenario de un fondo submarino

Modelos utilizados. En esta aplicación se muestran modelos de nubes de puntos de corales, capturados en su entorno y con ellos se recrea ese escenario, integrando en él más modelos tanto capturados en entornos reales como elementos virtuales creados.

- **Modelos capturados en entornos reales.** Para este demostrador los modelos capturados en entornos reales utilizados son los corales que se pueden ver en la Figura 4.1.
- **Elementos virtuales.** El resto de modelos del escenario, que se pueden ver en las Figuras 4.2, 4.3 y 4.4, son distintos tipos de peces y animales marinos, pequeñas plantas y los restos de un naufragio. Estos son modelos descargados de Sketchfab y no han sido tomados en entornos reales.

Características del demostrador. Para construir este demostrador se han utilizado algunas de las herramientas explicadas en la sección 3.2 del Capítulo 3.

Todos los modelos integrados, independientemente de su procedencia, han sido importados como se explica en la sección 2.3 y cargados como se detalla en el apartado "Carga de modelos 3D" de la sección 3.2. Siguiendo los apartados "Creación del entorno: Entorno natural submarino" y "Efecto agua" de esa misma sección se han creado el terreno y la ambientación, directamente en

Unreal. Además los modelos de peces introducidos, se han animado para que se muevan continuamente por el escenario, de la manera en la que se explica en el apartado "Trayectorias predefinidas" de la sección 3.2. Aunque estas trayectorias son realizadas una sola vez, se le ha añadido un bucle para que realice esa misma ruta continuamente. En este programa no es necesario marcar la opción para simular las interacciones físicas ya que las mallas estáticas de los corales y las plantas están colocadas en el fondo marino, mientras que los peces se mueven por encima y con una trayectoria fija que repiten constantemente, de forma que no impactan ni contra el resto de elementos del escenario ni entre ellos. Lo que sí se desactiva es la gravedad para que cree un efecto de que los elementos del escenario flotan. Tampoco se utilizan las entradas desde los controladores u otros periféricos ya que esta aplicación está diseñada solamente para mostrar, en un museo de ciencias naturales por ejemplo, una escena de una zona de difícil acceso como es el fondo marino y los arrecifes de coral. Conociendo y viendo lo que allí hay como si se estuviese en ese lugar y paseando por él sin dificultades ni peligro y además sin deteriorar ese escenario real. No está pensada para interactuar con los elementos del demostrador como si de un videojuego se tratase, solo para contemplarlos y aprender sobre ese escenario.

Resultados. Este demostrador se ha creado sobre todo pensando en una de las aplicaciones que actualmente más está avanzando en el terreno de la realidad virtual, su uso en museos o centros de enseñanza para propiciar la divulgación científica. Gracias a la realidad virtual es posible reconstruir en 3D objetos y lugares de interés o históricos, o lugares inaccesibles como es el caso del escenario del fondo submarino desarrollado en este proyecto. Este permite ver y conocer zonas únicas por su flora y su fauna de una manera realista, que de otro modo no se podrían mostrar al público y no se podrían estudiar con tanto detalle. Además se llama la atención incluso de los más jóvenes al utilizar las nuevas tecnologías en ambientes como los museos, de forma que se despierta su interés en la cultura y en probar nuevas experiencias para aprender. La apariencia y resultado final de la aplicación se muestra en la Figura 4.5.

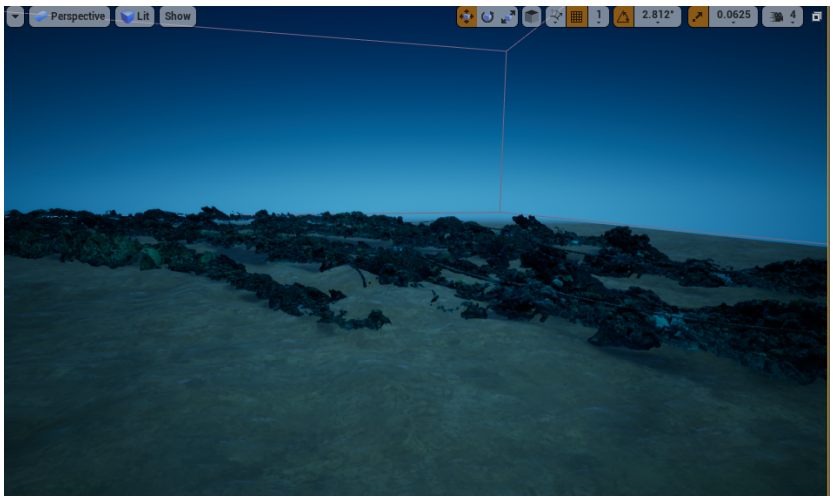
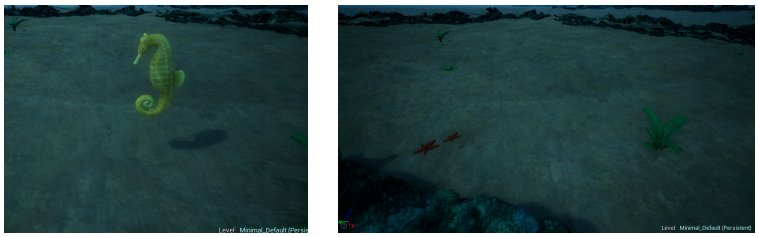


Figura 4.1: Modelos de corales capturados en un entorno real.

Figura 4.2: Modelos de distintos tipos de peces.



(a) (b)

Figura 4.3: Modelos de distintos elementos virtuales integrados dentro del escenario creado. (a) Caballito de mar. (b) Diferentes plantas y estrellas de mar.

Figura 4.4: Modelo de los restos de un naufragio integrado en el escenario.

Figura 4.5: Dos capturas realizadas durante la ejecución del programa en las que se ven distintas partes del escenario submarino final.

4.2. Demostrador en escenario de interiores

Modelos utilizados. Esta aplicación incorpora otro tipo de modelos reales capturados desde una fuente distinta, el *structure sensor*.

Los **modelos de mallas 3D capturados en entornos reales** que se han integrado en este demostrador son:

- Un modelo de la puerta del laboratorio en el que se ha desarrollado el proyecto como ejemplo de captura de una zona de una habitación, que se puede ver en la Figura 4.6.
- Modelos de un robot, las gafas de realidad virtual utilizadas en este proyecto y una hucha, como ejemplos de objetos de diferentes tamaños que pueden ser capturados. Estos tres aparecen en la Figura 4.7.

Por otro lado, los **elementos virtuales** que se han utilizado son:

- El suelo, las paredes y el techo de la estancia, realizados con las herramientas de Unreal Engine. Esto se puede observar en la Figura 4.8.
- Modelos de ventanas y de elementos de oficina como mesas, sillas y ordenadores, descargados desde Sketchfab y que se pueden ver en la Figura 4.9.

Características del demostrador. De igual manera que en la otra aplicación, se han utilizado algunas de las herramientas explicadas en la sección 3.2 del Capítulo 3, para la creación de este escenario.

La habitación que se ha recreado se ha construido directamente en Unreal Engine, siguiendo los pasos del apartado "Creación del entorno: Escenario de interior" de la sección 3.2. Tanto los elementos virtuales como los tomados en un escenario real se han importado a la escena de igual manera, como se explica en el apartado "Carga de modelos 3D" de la sección 3.2. En este demostrador se ha incluido el modelo de un robot que es movido por el usuario, lo que se consigue siguiendo las instrucciones del apartado "Entradas desde mandos o teclado" de la sección 3.2. En esta aplicación se ha implementado esta característica, a diferencia de en la aplicación descrita en la sección anterior, ya que esto puede permitir hacer pequeñas simulaciones para ver si se pueden introducir otros robots o sistemas móviles en el espacio de interior que se esté estudiando y si estos pueden moverse con libertad por las zonas que se quieran. De esta manera, el modelo del sistema a introducir puede ser movido por el usuario por todos los lugares por los que deberá moverse en el espacio real. También se ha marcado la opción para simular interacciones físicas, debido a que si el robot está moviéndose por el escenario no debe traspasar el resto de elementos en él, ya que lo que se quiere es que todos los elementos en el escenario tengan un comportamiento lo más próximo a la realidad posible.

Resultados. En este demostrador se ha implementado un escenario de interiores, donde se han integrado otro tipo de modelos del entorno capturados por sistemas inteligentes, los modelos de mallas 3D. Estos sistemas pueden tomar modelos en 3D tanto de objetos como de espacios de interior, en los cuales luego se puede navegar para inspeccionarlos en más detalle y poder realizar tareas

de mantenimiento o pequeñas simulaciones en ellos. De esta manera se pueden llevar a cabo controles e inspecciones de estos espacios, por ejemplo en industrias, sin necesidad de estar ahí y de manera sencilla ya que se pueden mirar los elementos del escenario desde cualquier distancia, ángulo y perspectiva, algo que no se puede hacer en la realidad. Esto puede ayudar en los controles de calidad y tareas de mantenimiento en una industria, así como en simulaciones para comprobar si se pueden introducir otro tipo de robots o maquinarias que se muevan sin colisionar y viendo si se posee el espacio suficiente para esto. El resultado final del escenario se puede ver en la Figura 4.10.

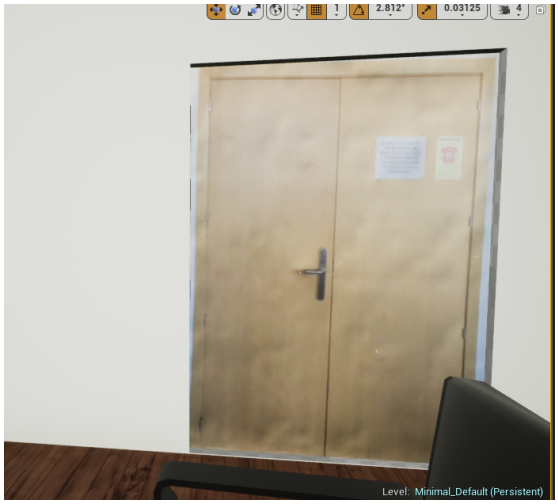


Figura 4.6: Modelo capturado con el *structure sensor* gracias a la App *Room Capture* mostrado dentro de la aplicación desarrollada.

(a) (b) (c)

Figura 4.7: Modelos de objetos capturados en un entorno real con el *structure sensor* gracias a la App *Scanner*, todos ellos mostrados dentro de la aplicación desarrollada. (a) Modelo de un robot. (b) Modelo de las gafas de realidad virtual. (c) Modelo de una hucha.

Figura 4.8: Paredes y suelo de la habitación creados en Unreal Engine que constituyen el escenario básico del segundo demostrador.

(a) (b)

(c) (d)

Figura 4.9: Elementos virtuales descargados de Sketchfab dentro de la aplicación. (a) Modelo de una ventana. (b) Mesa con pequeños modelos de objetos de oficina. (c) Sillas y mesas dentro del escenario. (d) Modelo de una pantalla y un teclado de ordenador.

(a) (b)

Figura 4.10: Dos capturas realizadas viendo distintas partes del escenario de interior final durante la ejecución del programa.

4.3. Manual de puesta en marcha

El procedimiento a seguir para poder visualizar las aplicaciones desarrolladas en las gafas de realidad virtual es el mismo para las dos.

- Lo primero que hay que hacer es conectar las gafas de realidad virtual y todos sus accesorios (estaciones base y controladores en caso de que sean necesarios) al ordenador. La conexión de todos los elementos viene explicada paso a paso en la sección A.3 del anexo A.
- Después se inicia el software SteamVR donde se puede ver si la conexión se ha realizado correctamente y el *headset* está preparado para ser utilizado. Aparecerá la ventana que se muestra en la Figura 4.11.
- Una vez esté listo esto, se abre el motor Unreal Engine y el proyecto que se quiera visualizar.
- Se recompila y construye el proyecto para que todos los elementos y los efectos de luces y ambientación se vean después correctamente.
- Se despliega la flecha que hay al lado de la herramienta *Play* y se elige el modo *VR Preview*. Esto se puede ver en la Figura 4.12.

Tras llevar a cabo este sencillo proceso, solo falta colocarse las gafas de realidad virtual y sumergirse en los escenarios creados.

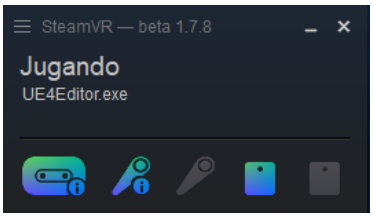


Figura 4.11: Ventana de SteamVR que muestra los elementos del *headset* conectados.

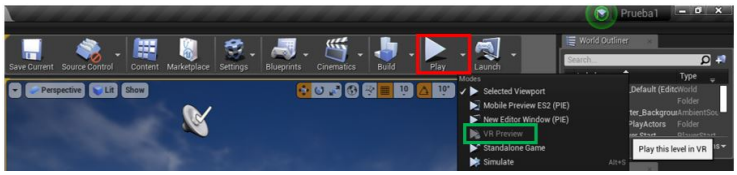


Figura 4.12: Opción *VR Preview* que hay que seleccionar en Unreal Engine para poder ver en las gafas de realidad virtual la aplicación desarrollada.

Capítulo 5

Conclusiones

Este capítulo presenta las conclusiones extraídas durante la realización de este proyecto, así como los problemas abordados en él. También se expone el estudio de las posibles mejoras que se pueden realizar partiendo de este proyecto y trabajos futuros que tomen este como base o que estén relacionados con él.

5.1. Conclusiones técnicas

El objetivo principal de este proyecto era poner en marcha un *headset* de realidad virtual y aprender tanto su funcionamiento como la manera de desarrollar una pequeña aplicación utilizando las herramientas diseñadas para ello. Este objetivo se ha cumplido, logrando aprender a utilizar de forma básica un hardware y un software con el que no se había trabajado y experimentado antes. Se ha estudiado la documentación y los sistemas más conocidos actualmente, lo que ha servido como toma de contacto con estas tecnologías y ha permitido sentar la base para alcanzar todos los objetivos propuestos. Como conclusión de esta parte, se ha comprobado el gran potencial de las herramientas elegidas, que tras la realización del proyecto, podemos concluir que resultaron muy adecuadas para el mismo.

El siguiente objetivo consistía en el uso de dos tipos de datos distintos capturados en entornos reales procedentes de dos fuentes diferentes, con los cuales se han implementado dos demostradores trabajando con un software específico y actualmente muy utilizado, *Unreal Engine*. Este segundo objetivo se ha cubierto satisfactoriamente creando dos demostradores de RV. Uno de ellos utiliza modelos de nubes de puntos, cedidos por un grupo de investigación colaborador del laboratorio de investigación en el que se realiza este trabajo. El otro demostrador utiliza modelos de mallas 3D, tomados directamente durante la realización de este proyecto con el *structure sensor* del que también se ha aprendido su modo de uso. Para realizarlos, se han estudiado las características de estos modelos y sus métodos de captura, sobre todo del *structure sensor* y sus aplicaciones para iOS ya que ha sido uno de los elementos utilizados en el proyecto, así como las maneras de importarlos al software utilizado. Para la creación de los demostradores se ha investigado el manejo del motor gráfico *Unreal Engine* y sus herramientas básicas. Se han conseguido implementar las características básicas deseadas, que incluyen: las interacciones físicas entre elementos de la escena, el

manejo de personajes con entradas desde el teclado, la realización de trayectorias predefinidas por parte de algunos personajes y la creación y modificación general del escenario y del ambiente de este logrando recrear dos entornos muy diferentes en cada aplicación. Al hacerlo, se ha visto que las interacciones físicas y que la creación/modificación de un escenario resulta fácil gracias a todas las opciones que ofrece el motor gráfico. En cambio, integrar características para animar e interactuar con las entidades cargadas en la escena como la creación de una ruta fija o la lectura de una entrada y la posterior reacción del personaje con ella resulta más complicado. En este proyecto no se llega a explotar todo el potencial ya que es una herramienta muy compleja que se usa en entornos profesionales, por lo que se puede seguir explorando en proyectos futuros y mejorando la base creada en este trabajo, como se expone en la sección 5.4.

Como conclusión más general, los demostradores construidos servirán como base para los ejemplos de divulgación y presentación del grupo de investigación, a la hora de mostrar los modelos del entorno capturados por distintas plataformas robóticas utilizadas habitualmente en él.

5.2. Conclusiones personales

Personalmente, estoy satisfecha con el trabajo ya que me ha servido como experiencia para ver como se trabaja en un laboratorio de investigación estando rodeada de personas con más experiencia de las que he aprendido y que me han ayudado y aconsejado.

Elegí este trabajo por la curiosidad de conocer más a fondo un campo tan en auge en la actualidad como la realidad virtual/aumentada y de aprender el manejo de alguna herramienta totalmente nueva y diferente a lo utilizado durante la carrera y creo que está decisión ha sido muy adecuada ya que me ha permitido lograr esto y ver como es enfrentarme al reto de aprender por mi misma algo nuevo y desconocido para mi. A pesar de los problemas encontrados, esto me ha servido para madurar y aprender mucho más que si hubiera realizado un trabajo con sistemas que conociera mejor de antemano. Gracias a este proyecto sé como enfrentarme y abordar retos así de manera más adecuada y adquiriendo los máximos conocimientos posibles, además de haber aprendido el uso básico de herramientas como el *headset* de realidad virtual y el programa *Unreal Engine* que me puede servir en un futuro.

5.3. Problemas encontrados

Durante la realización de este proyecto se han tenido que abordar diversos problemas. El primero fue el aprender a buscar documentación y publicaciones científicas en el ámbito de la investigación académica y analizar esta información.

Una vez realizada la investigación y tras adquirir los conocimientos básicos necesarios para empezar, durante el desarrollo de la parte práctica, aparecieron problemas derivados de la falta de experiencia en el uso de las herramientas utilizadas. En concreto, por un lado aparecieron problemas en la conexión adecuada del *headset*, debido a que al principio no se estaba conectando el cable *DisplayPort* en la tarjeta gráfica dedicada del equipo ya que las tarjetas integradas (conectadas a la placa base del equipo) no tienen la potencia suficiente

para usar Vive Pro. Por el lado del uso del software surgieron errores típicos al utilizar un nuevo entorno de trabajo al implementar alguna de las funcionalidades que se han estudiado en Unreal Engine, que se solucionaron tras buscar en su extensa documentación y gracias a la gran comunidad existente de personas que desarrollan aplicaciones con estas herramientas.

Un importante inconveniente a la hora de realizar proyectos de este tipo es la necesidad de un ordenador potente con una buena tarjeta gráfica que soporte el uso del *headset* de realidad virtual, en este caso Vive Pro, y que permita que el motor gráfico, *Unreal Engine*, funcione de manera fluida. Los requerimientos necesarios para el buen funcionamiento del sistema están plasmados en la sección A.1 del anexo A.

5.4. Trabajo Futuro

Como se ha comentado en los apartados anteriores, este proyecto es solo una base que se puede continuar desarrollando y profundizar más en ella. Hay diferentes posibilidades que se pueden seguir en un futuro y que se han ido estudiando y planteando durante la realización de este trabajo.

La primera alternativa que se ha pensado es la integración de reconocimiento en los demostradores desarrollados. El reconocimiento de objetos puede ser útil tanto en la aplicación de escenarios de interiores como en la aplicación destinada a su uso en museos. En el primer caso, además de navegar por el escenario para realizar tareas de control, mantenimiento o simulaciones, se podrían ver los objetos, maquinarias o robots que en él estuvieran y conocer cual es cada uno para los controles correspondientes. En el segundo caso, además de introducirte en el escenario y ver el fondo submarino con sus especies típicas se podría además mostrar cuál es cada coral y cada animal u objeto que aparezcan en él, algo muy conveniente en el demostrador de un museo. Se podría incluso mostrar información de lo que se está viendo, yendo un poco más allá en esta mejora.

Otro de los caminos que se puede seguir, es la explotación de todas las posibilidades que ofrece el *headset* Vive Pro, como por ejemplo incorporar el uso de las cámaras estéreo que posee el visor, que son una de las mejoras respecto a su versión anterior, las HTC Vive. Se describen detalles sobre su funcionamiento en el anexo A. Con ellas se puede ver la habitación en la que te encuentras, lo que podría servir para crear nuevas experiencias mezclando lo que se ve a través de las cámaras frontales y el mundo virtual, creando alguna aplicación de realidad mixta o modificando alguna de las dos creadas añadiéndole la funcionalidad de poder alternar entre el mundo virtual y el real.

Apéndice A

Headset HTC Vive Pro

Este anexo describe los componentes, con sus cables y sensores, de manera detallada, así como las especificaciones del *headset*. También se explican los fundamentos básicos de su sistema de seguimiento, parte esencial para una buena experiencia de realidad virtual. Finalmente se proporciona un manual para la conexión del *headset* al ordenador paso por paso.

A.1. Componentes y especificaciones

A.1.1. Componentes y sensores

En la caja del *set* completo de realidad virtual se encuentran los siguientes accesorios.

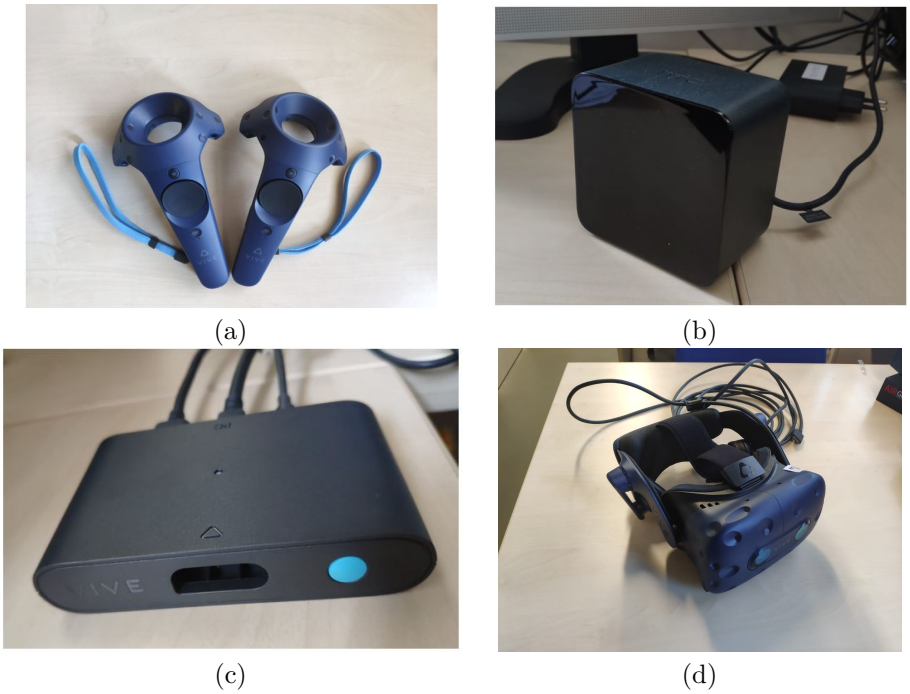
Controladores. Hay dos controladores inalámbricos con cordones para una mejor sujeción a la mano. Los controladores tienen unos pequeños huecos que es donde se encuentran los sensores de posicionamiento láser necesarios para el sistema de seguimiento. Además de esto vienen dos cables micro-USB con dos adaptadores de corriente para poder cargar ambos mandos.

Estaciones Base. Hay dos estaciones base, indispensables para poder realizar el seguimiento tanto del visor como de los controladores. También se tienen dos cables de alimentación, uno para cada una y el *kit* de montaje para poder colocar las estaciones base en la pared de la habitación que se convertirá en el área de juego, donde hay que colocarlas de manera que abarquen todo el área.

Caja *LinkBox*. Se dispone de esta caja, necesaria para la conexión del HMD al ordenador, equipada con un paño de limpieza y dos tapas para los orificios de los auriculares. Se encuentran también los 3 cables que irán conectados desde la *LinkBox* hasta el ordenador: el cable de alimentación, el cable *DisplayPort* y un cable USB 3.0.

Gafas de realidad virtual. Se tiene el visor que viene con tiras ajustables y almohadillas que se adaptan mejor a la cara para una mayor comodidad cuando te lo pones y para lograr que entre menos luz exterior al visor. De igual modo

que los controladores, está lleno de huecos donde se encuentran los sensores de posicionamiento, en este caso un total de 37. El visor no es inalámbrico, de él sale un cable que se conecta a la *LinkBox*. También tiene incorporados auriculares y micrófonos dobles con cancelación de ruido, que permiten una experiencia más inmersiva. Posee además dos cámaras estéreo frontales que permiten mostrar un amplio campo visual del entorno real en el que se está, que realizan una percepción 3D y detección de profundidad con unos sensores estéreo RGB. Esta cámara puede ser activada por el usuario pulsando dos veces el botón *System* de *SteamVR* o automáticamente cuando el usuario se acerque demasiado a los límites del área de juego trabajando conjuntamente con *Chaperone* (sistema incorporado para la seguridad mientras se utilizan las gafas), creando otra capa que mantiene al jugador seguro mientras se mueve utilizando el visor. Además de poder acceder a las imágenes en bruto se puede acceder también a la profundidad, mapeo espacial (mallas estáticas y dinámicas) y módulo de visión AI para la segmentación semántica, permitiendo colocar objetos virtuales, interacciones con estos objetos e interacciones manuales simples. Esto es gracias los módulos que tiene: un módulo de profundidad, un módulo de acceso directo, un módulo de reconstrucción 3D y un módulo de visión AI. Hay soporte para el desarrollo con plugins para *Unity* y *Unreal Engine*.



(a) (b) (c) (d)

Figura A.1: Componentes principales del *headset* completo HTC Vive Pro. (a) Controladores. (b) Estación Base. (c) *Linkbox*. (d) Gafas de realidad virtual.

A.1.2. Especificaciones

Especificaciones del *headset*.

- **Tipo**: RV/RA *Head-mounted display*.
- **Desarrolladores**: HTC, Valve.
- **Plataforma**: SteamVR.
- **Pantalla**: Dual AMOLED 3.5" diagonal.
- **Resolución**: 2880 x 1600 (1440 x 1600 por ojo).
- **Densidad de píxeles**: 615 PPI (*pixels per inch*) por ojo.
- **Frecuencia de actualización**: 90 Hz.
- **Campo de visión**: 110º.
- **Conexiones**: USB 3.0, *DisplayPort* 1.2, *Bluetooth*.
- **Seguimiento**: 6GDL.
- **Sensores**: giroscopio, acelerómetro, seguimiento SteamVR, proximidad.
- **Cámara**: Dos cámaras estéreo frontales.
- **Entrada**: Micrófonos dobles, cancelación activa de ruido y modo alerta o modo conversación.
- **Audio**: Auriculares con certificado de alta resolución (extraíbles).
- **Ergonomía**: Distancia de la lente, IPD (distancia interpupilar), auriculares y tiras de la cabeza ajustables.
- **Área de juego mínima**: 2m x 1.5m.
- **Área de juego máxima**: 6m x 6m.
- **Precio**: 1.399€(el *set* completo), 879€(solo Vive Pro HMD).

Requerimientos del sistema. Es necesario un ordenador con unas ciertas características para que el *headset* funcione de forma adecuada. Los requisitos mínimos recomendados son los siguientes:

- **Sistemas operativos compatibles**: *Windows* 8 o *Windows* 10.
- **RAM**: 4 GB o superior.
- **CPU**: Procesador Intel Core i5 4590 o AMD FX 8350 o superior.
- **GPU**: NVIDIA GeForce GTX 1060 o superior o Radeon RX 480 o superior.
- **Puertos**: Un puerto USB 3.0 y un puerto de vídeo *DisplayPort* 1.2.

A.2. Sistema de seguimiento: *Lighthouse* o faro

Este sistema de seguimiento y posición desarrollado por Valve Software se encuentra tanto en el visor como en sus controladores que están llenos de sensores y son las estaciones base el elemento activo que permite el seguimiento.

Este es un sistema de seguimiento de escaneo de haz que utiliza barridos de luz programados que golpean a los sensores dispuestos en una configuración conocida para calcular los 6 GDL de la posición de los objetos. El funcionamiento básico de este sistema consiste en que uno o varios emisores realizan un barridos de luz láser periódicamente, una vez en horizontal y después una vez en vertical. Un pulso de luz emitido por un grupo de LED es lanzado de manera sincronizada con el comienzo de cada ciclo de barrido láser. Esto se produce con una frecuencia de 60 veces por segundo. Tanto en el visor como en los controladores hay numerosos foto-sensores colocados cada uno de los cuales sirve para calcular el tiempo de retraso entre el destello de sincronización de los LED hasta que reciben el haz del barrido, primero del horizontal y luego del vertical. Con esto y la velocidad conocida del barrido láser, el sistema puede determinar el ángulo con el que el haz impactó en cada sensor y con el conjunto de ángulos de cada uno de ellos se puede determinar la posición y orientación del objeto. Con esta técnica se obtienen los 6 GDL de forma precisa y rápida y además como mejora añadida HTC Vive Pro utiliza IMU's, un conjunto de acelerómetros y giroscopios, de manera que al combinar las dos técnicas la precisión mejora.

Los elementos con los que está construido este sistema son sencillos, con pocos componentes se obtienen estos precisos resultados. Se construye con una placa estática con 9 LED para emitir el destello de sincronización y dos tambores giratorios, uno para el barrido láser horizontal y otro para el barrido láser vertical, que poseen partes móviles que permiten la rotación del haz láser.

Una de las ventajas de este sistema radica en que aunque se incremente el número de unidades rastreadas no se afecta al tiempo total requerido para rastrear todas las unidades, ya que los sensores se encuentran en el objeto a seguir, además de su precisión y rapidez ya comentada. Como inconveniente de este método es su latencia inherente debido a que utiliza el tiempo como medio para hacer los cálculos y también la necesidad de que no interfiera ningún objeto entre el láser y el sensor.

A.3. Conexión al ordenador y puesta en marcha

Conexión *LinkBox*-ordenador. El visor es el único elemento que debe estar conectado al ordenador. Dicha conexión se realiza a través de la *LinkBox* de la siguiente manera:

- En *set* vienen 3 cables junto con la *LinkBox*: el cable de alimentación, el cable *DisplayPort* y un cable USB 3.0. Estos se han de poner en el conector propio de cada uno de la *LinkBox*.
- Se conecta el USB 3.0 a un puerto USB adecuado disponible del ordenador.
- Se conecta el cable *DisplayPort* a la tarjeta gráfica dedicada del ordenador. Este cable debe conectarse a la misma tarjeta gráfica a la que está conectada el monitor.

- Se conecta el cable de alimentación a una toma de corriente.

La conexión de los cables a la *Linkbox* y al ordenador se puede ver en la Figura A.2.

(a) (b)

Figura A.2: Conexión entre la *Linkbox* y el ordenador. (a) Cable USB 3.0, *DisplayPort* y alimentación conectados a la *Linkbox*. (b) Conexión al ordenador del USB 3.0 en rojo y del *DisplayPort* en amarillo.

Conexión visor-*LinkBox*. Esta conexión es muy sencilla, basta con conectar el cable que sale de las gafas al conector que queda libre para este propósito y pulsar el botón azul de encendido que hay a su lado, que se puede ver en la Figura A.3.

Figura A.3: Puerto para conectar la *Linkbox* y las gafas y su botón azul de encendido.

Puesta en marcha de las estaciones base. Hay que encenderlas y sincronizadas una vez hayan sido colocadas de manera adecuada y acorde con el área de juego elegida para usar el *headset*.

- Se colocan las estaciones base sobre superficies estables o con el *kit* que se proporciona en esquinas opuestas del área de juego, diagonalmente, de manera que ambas miren al centro del área cubriendo así la totalidad del espacio.
- Es adecuado que estén a 2m o más de altura e inclinadas un ángulo de 30º-45º.
- Se conectan los cables de alimentación a las estaciones base y estos a tomas de corriente, tras lo que la luz de estado de las estaciones base se encenderá en verde.
- Se sincronizan y se eligen los canales de las estaciones base presionando los botones de la parte trasera de estas.
- En una estación base se selecciona el canal "b" mientras que en la otra se elige el "c", si no se tiene cable de sincronización.
- En una estación base se selecciona el canal "a" mientras que en la otra se elige el "b", si se tiene cable de sincronización.
- Una vez colocadas se ajusta el área de juego desde SteamVR haciendo *click* en *Run Room Setup*.
- Después de este proceso no se deben mover las estaciones base, de lo contrario habrá que ajustar el área de juego de nuevo.

Encendido y conexión de los controladores. Los controladores son inalámbricos de manera que no hay que conectar ningún cable físicamente, los pasos a seguir para que sean reconocidos por el ordenador y poder ser utilizados son los siguientes:

- Cargar ambos mandos ya que vienen descargados inicialmente.
- Encender los controladores pulsando su botón de sistema.
- Abrir SteamVR comprobando que el visor y las estaciones base están conectados, hacer *click* derecho en el icono de los mandos y seleccionar *Pair controller*, donde te vendrán las siguientes instrucciones.
- Pulsar a la vez el botón de sistema y el botón de menú del controlador, que se pueden ver en la Figura A.4, hasta que parpadee la luz de estado en color azul.
- Una vez emparejados esa luz será verde y fija y aparecerán como conectados y activos en la pantalla de SteamVR.

Figura A.4: Controladores, con su botón de sistema rodeado en amarillo y su botón de menú rodeado en rojo.

Apéndice B

Proceso detallado de creación de una trayectoria predefinida en Unreal Engine 4

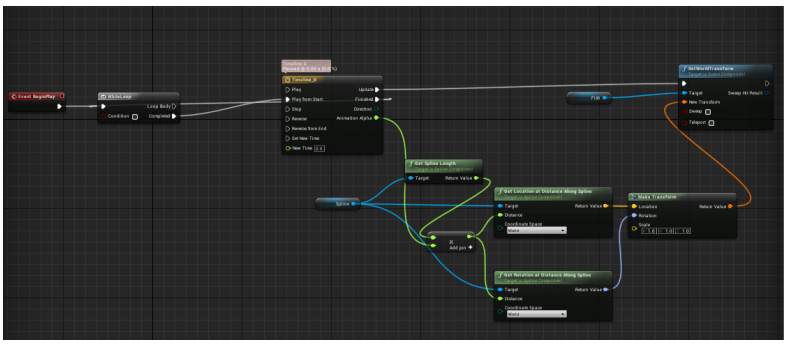
Para lograr que un elemento en un escenario creado en Unreal Engine 4 siga una trayectoria predefinida sin necesidad de ser controlado por el usuario, es necesaria la creación de un componente llamado *Spline*. Para ello se deben seguir los siguientes pasos:

- Abrir el editor del personaje que se quiera que siga la ruta haciendo doble *click* sobre él.
- En la pestaña *Viewport* de la ventana que se acaba de abrir pulsar en *Add Component* y añadir el componente *Spline*.
- Abrir la pestaña *Event Graph* donde se va a programar el seguimiento de la trayectoria.
- Se de al botón derecho del ratón, se busca un bloque *Timeline* y se coloca en la pantalla.
- Se une el evento *Event Begin Play* (que se activa automáticamente al empezar la simulación) con la entrada *Play from Start* del bloque *Timeline*.
- Se hace doble *click* en el bloque *Timeline* para abrir su editor.
- En este editor se hace *click* en el botón en que aparece una *f* en la esquina superior izquierda para crear una nueva pista.
- En la variable *Length* se escribe el tiempo que se desea que tarde en realizar la trayectoria.
- Se hace *click* derecho y se selecciona la opción *Add key to CurveFloat_0*.

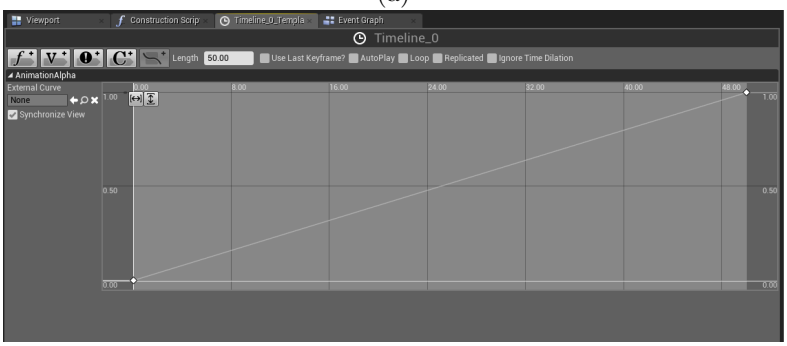
- Se cambian sus variables *Time* y *Value*, dándoles un valor inicial de 0.0 a ambas.
- Se pueden ir añadiendo todos los puntos que se deseen en diferentes instantes de tiempo y con diferentes valores, en este caso solo se ha añadido uno más en el instante final y con un valor de 1.0.
- Se vuelve a la pantalla de *Event Graph* y se coloca un bloque *SetWorldTransform* que es el que hará que la localización del personaje se actualice siguiendo la ruta fijada.
- Se arrastra desde la parte izquierda de componentes el personaje que se quiera que siga la trayectoria, en este caso *Fish* hasta la pantalla de *Event Graph* y se conecta a la entrada *Target* del bloque *SetWorldTransform*.
- Se conecta la salida *Update* del *Timeline* con la entrada del bloque *SetWorldTransform*.
- Se coloca el bloque *Get Spline Length*.
- Se arrastra a la pantalla el componente *Spline* para que sea la referencia del bloque que se acaba de colocar, conectándolo a su entrada *Target*.
- Se coloca un bloque $float * float$ y a una de sus dos entradas se conecta la salida *Return Value* del bloque *Get Spline Length*.
- Se conecta la salida *Animation Alpha* (en este caso, por defecto el nombre es *NewTrack_0*) del bloque *Timeline* a la otra entrada del bloque $float * float$, de esta manera como la pista del *Timeline* va de 0.0 a 1.0 se irá recorriendo la longitud de la trayectoria de principio a fin.
- Se colocan dos bloques más *Get Location at Distance Along Spline* y *Get Rotation at Distance Along Spline*.
- En ambos se conecta el componente *Spline* a su entrada *Target* y en *Coordinate Space* se selecciona *World*.
- Conocida la distancia se quiere saber la localización y la rotación del personaje durante su trayectoria, para ello en la entrada *Distance* de los dos bloques que se acaban de colocar se conecta la salida del bloque $float * float$.
- Se coloca un bloque *Make Transform*, cuya salida se conecta a la entrada *New Transform* del bloque *Set World Transform*, que es el que hace que tanto la posición como la orientación del pez vayan actualizándose.
- En las entradas *Location* y *Rotation* del bloque *Make Transform* se conectan las salidas de los bloques *Get Location at Distance Along Spline* y *Get Rotation at Distance Along Spline* respectivamente, de manera que la transformación que se de al personaje sea la fijada por la trayectoria *Spline*.
- Se compila y se cierra este editor, yendo de nuevo a la ventana principal del escenario donde se modela la trayectoria a seguir como se desee.

- Se hace *click* con el botón derecho, se selecciona la opción *Visualize Roll and Scale* y se van añadiendo puntos a la ruta desde los cuales se puede mover, girar y estirar la trayectoria hasta lograr el resultado que se quiera.
- Ya se tiene la trayectoria predefinida, si se da al *Play* se ve como el personaje va siguiéndola.

Estas instrucciones sirven para crear una trayectoria que el personaje seguirá una sola vez y cuando la finalice se quedará parado. Si se quiere que repita la ruta continuamente se ha de añadir un bucle *While* justo después del evento *Event Begin Play* del que se conecta su salida a la entrada del bucle. La salida *Completed* del bucle se conecta a la entrada *Play from Start* del *Timeline*, donde antes estaba conectado el evento de inicio. Finalmente se conecta la salida *Finished* del bloque *Timeline* a la entrada del bucle *While* de manera que cada vez que se termine una trayectoria vuelva a empezar. En la Figura B.1 se muestra como queda el *Event Graph*, también el editor del bloque *Timeline*, además de como se ve la trayectoria creada en la pantalla principal.



(a)



(b)

(c)

Figura B.1: Resultados finales tras la creación de la trayectoria. (a) *Event Graph* con los bloques necesarios para crear una trayectoria cíclica. (b) Editor del bloque *Timeline*. (c) Apariencia final de la trayectoria de un personaje en el escenario.

Bibliografía

[1] Epic Games. Unreal engine 4 documentation. *URL https://docs. unrealengine. com/latest/INT/index. html*, 2019.

[2] YAGV Boas. Overview of virtual reality technologies. In *Interactive Multimedia Conference*, volume 2013, 2013.

[3] HTC VIVE. *HTC Vive Pro HMD user guide*, 2019.

[4] Diederick C Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205, 2017.

[5] Xavier Basogain, Miguel Olabe, Koldobika Espinosa, C Rouèche, and JC Olabe. Realidad aumentada en la educación: una tecnología emergente. *Escuela Superior de Ingeniería de Bilbao, EHU. Recuperado de http://bit. ly/2hpZokY*, 2007.

[6] Jorge Trindade, Carlos Fiolhais, and Leandro Almeida. Science learning in virtual environments: a descriptive study. *British Journal of Educational Technology*, 33(4):471–488, 2002.

[7] Julia Hayes and Kyungjin Yoo. Virtual reality interactivity in a museum environment. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, page 131. ACM, 2018.

[8] T. M. Bonanni G. Grisetti L. Iocchi D. Nardi C. Stachniss J. Serafin, M. Di Cicco and V. A. Ziparo. Robots for exploration, digital preservation and visualization of archeological sites. *Artificial Intelligence for Cultural Heritage*, 2016. URL http://www.rovina-project.eu/.

[9] Santiago González Izard, Juan A Juanes Méndez, Pablo Ruisoto Palomera, and Francisco J García-Peñalvo. Applications of virtual and augmented reality in biomedical imaging. *Journal of medical systems*, 43(4):102, 2019.

[10] Fausto Bernardini and Holly Rushmeier. The 3d model acquisition pipeline. In *Computer graphics forum*, volume 21, pages 149–172. Wiley Online Library, 2002.

[11] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[12] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[13] BORJA JAVIER HERRÁEZ CONCEJO. Segmentación y clasificación de mallas 3d. 2016.

[14] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Röossl. Geometric modeling based on polygonal meshes. 2007.

[15] Microsoft. Guía de paint. *http://ensinobasico.epapontevedra.com/resources/GUIA+de+Paint.pdf*, 2019.

[16] William R Sherman and Alan B Craig. *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann, 2018.